

Цель: создать цифровую модель сигнальной лампы, которая будет переключать свои состояния аналогично реальному уличному светофору.

Планируемые результаты:

- знать основные понятия микроконтроллеров и их роль в управлении электронными устройствами
- знать структуру программы управления сигнальной лампой и правила написания программного кода
- уметь запрограммировать микроконтроллер для управления световыми эффектами сигнальной лампы

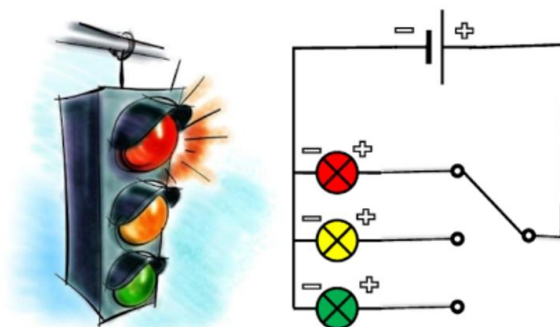
Используемое оборудование и материалы:

- Компьютер с Arduino IDE (версия 1.8)
- Микроконтроллер DXL-IOT
- Преобразователь питания DXL-PWR для микроконтроллера
- Блок питания для микроконтроллера
- Светодиоды трехцветные
- Провода
- Корпусные детали для монтажа электроники



Теоретический материал

Принцип работы светофора



Светофор (англ. traffic light) - это светосигнальное устройство, предназначенное для регулирования движения транспортных средств и пешеходов через перекрестки и пешеходные переходы. Принцип работы светофора основан на использовании трех цветовых сигналов: красного, желтого и зеленого.

Красный сигнал светофора запрещает движение транспортных средств. Этот сигнал загорается первым, когда светофор находится в режиме ожидания. Красный свет должен быть виден всем участникам дорожного движения, чтобы они могли остановиться и подготовиться к движению.

Желтый сигнал светофора сообщает о предстоящем переключении сигнала светофора. Этот сигнал часто используется для предупреждения водителей о необходимости снизить скорость или остановиться перед переходом на зеленый свет. Желтый свет должен гореть около 3-5 секунд, а затем переключается на зеленый.

Зеленый сигнал светофора разрешает движение транспортных средств через перекресток. Этот сигнал обычно загорается последним, когда все транспортные средства остановлены и готовы к движению. Зеленый свет должен гореть в течение времени, достаточного для того, чтобы все транспортные средства успели проехать перекресток, но не дольше, чем это необходимо.

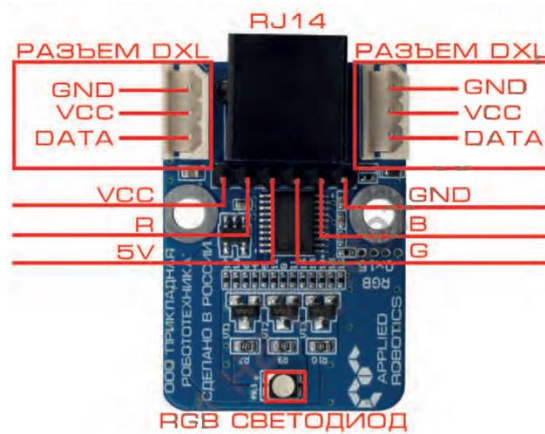
Индикационные модули: модуль звукового пьезоизлучателя, модуль светодиода, модуль трехцветного светодиода.

Светодиоды (LED, Light Emitting Diode) используются для индикации различных состояний устройства. Эти компоненты сверхяркие, экономичные и имеют долгий срок службы. Светодиоды управляются сигналами от микроконтроллера, который регулирует их включение и выключение.

Работа светодиодной индикации:

1. Микроконтроллер посылает сигнал на включение/выключение светодиода.
2. Сигнал приводит к тому, что через светодиод начинает течь электрический ток.
3. Светодиод излучает свет, индицируя определенное состояние устройства.

Модуль «Трехцветный светодиод» представляет собой стандартный RGB светодиод, размещенный на плате с микроконтроллером. Для управления каждым каналом в отдельности на модуле выведены отдельные линии. Данный модуль предполагается использовать для более удобного подключения такого индикационного элемента, как RGB-светодиод, к популярным контроллерам. Внешний вид модуля представлен на рисунке.



Модуль «Трехцветный светодиод» имеет:

- Разъем DXL - два трехпиновых разъема типа Molex, содержащих в себе линии GND (земля), VCC (питание), DATA (линия данных). Используются для подключения модуля по интерфейсу Dynamixel, как в одиночном виде, так и в составе цепи устройств.
- Разъем типа RJ14 для подключения модуля в фирменную плату расширения для подключения сенсорных модулей.
- Шестипиновый разъем.

Изменение значений ID модулей.

Для того, чтобы использовать несколько одинаковых периферийных модулей одновременно, например, два модуля «Светодиод», необходимо изменить их значения ID, сделав их разными. Это требуется сделать для однозначной идентификации модуля в цепи устройств. Если в цепи окажется более одного Dynamixel-совместимого устройства с одинаковыми ID (например, модуль «Светодиод» со значением ID 9 и сервопривод Dynamixel со значением ID 9), то возникнет ошибка при обмене данными, что может привести к полной неработоспособности устройств, находящихся в ней.

Для изменения ID периферийного модуля необходимо использовать «Универсальный вычислительный контроллер DXL-IOT», а также установленную библиотеку «DxlMaster2». Для работы с модулем на низком уровне необходимо использовать встроенный в библиотеку пример «Console2». После чего, подключив модуль к контроллеру, загрузить управляющий код из примера «Console2», открыть монитор порта и дождаться символа «>».

Далее необходимо выполнить команду «ping», чтобы убедиться, что связь с модулем корректно установилась. Данная команда может быть вызвана в следующем виде:

ping X, где X - значение ID модуля.

Получив ответ «ok» статуса установки соединения с периферийным модулем, можно изменить значение ID

write 9 3 10, где 9 – текущий ID, 3 – номер регистра, 10 – новое значение.

После чего следует выполнить перезагрузку модуля.

Практические примеры подключения модулей к контроллерам и примеры программ описаны в учебном пособии «Периферийные функциональные модули. Часть 1» (ООО «Прикладная робототехника»).

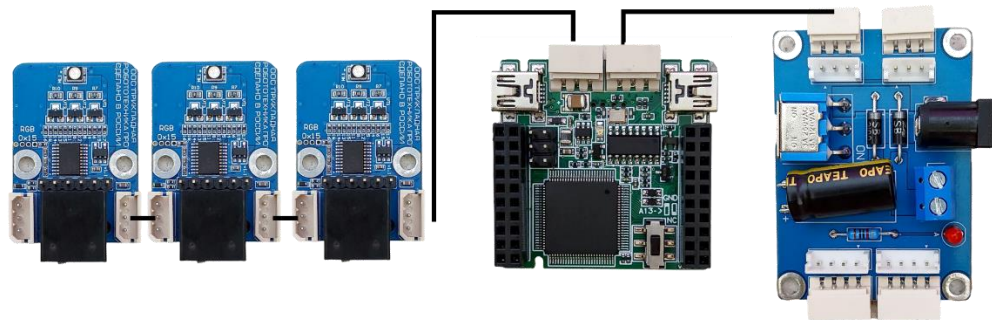
Практическая часть

Задания:

1. Собрать корпус светодифузора, используя детали и необходимую электронику из набора «Стем мастерская»



2. Назначить ID (при необходимости) и подключить светодиоды к микроконтроллеру
 Для проверки работоспособности датчиков можно воспользоваться скетчем *Console_v2* из примера библиотеки *DxlMaster2*. С помощью данного скетча можно изменять ID.



3. Написать программу для микроконтроллера, эмулирующую работу светофорного режима (за основу можно взять фрагмент кода из приложения)

Описание регистра модуля «Трехцветный светодиод»

ID	Наименование регистра	Адрес	Тип данных	Допустимый диапазон
0x15	GREEN_LED_DATA	26	uint8_t	0..255
	RED_LED_DATA	27		
	BLUE_LED_DATA	28		

4. Подключить микроконтроллер к компьютеру и загрузить программу
5. Провести тестирование работы светофора

Контрольные вопросы

1. Какой сигнал светофора сообщает о предстоящем переключении сигнала и какова его функция?
2. Как можно создать задержку между изменениями состояний светофора с использованием Arduino?
3. Какие регистры модуля "Трехцветный светодиод" необходимо использовать для управления зеленым, красным и синим светодиодами?
4. Какие принципы работы светофора необходимо учитывать при программировании микроконтроллера?
5. Как можно улучшить данную программу для управления светофором с использованием светодиодов?

Фрагмент кода

```
#include "DxlMaster.h"

const unsigned long dynamixel_baudrate = 57600;
const unsigned long serial_baudrate = 57600;
const uint8_t id1 = 1;
const uint8_t id2 = 2;
const uint8_t id3 = 3;
DynamixelDevice light1(id1), light2(id2), light3(id3);

void setup() {
  DxlMaster.begin(dynamixel_baudrate);
  light1.init();
  light2.init();
  light3.init();
  Serial.begin(serial_baudrate);
}

void loop() {
  TL(light1, 26, 255, 0);
  TL(light2, 26, 255, 0);
  TL(light3, 26, 255, 0);
}

void TL (DynamixelDevice light, int REG, int VAL1, int VAL2)
{
  light.write(REG, VAL1);
  delay(100);
  light.write(REG, VAL2);
  delay(100);
}
```

Цель: разработать интерактивную систему, позволяющей пользователю выводить текст при помощи кнопок и просматривать его на дисплее, а также отслеживать состояние с помощью светодиодов.

Планируемые результаты:

- знать структуру программы управления кнопками, светодиодами и дисплеем
- уметь программировать микроконтроллер для вывода текстового сообщения и индикации светодиодами в зависимости от нажатой кнопки

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер DXL-IOT
- Дисплей LCD
- Преобразователь питания DXL-PWR для микроконтроллера
- Блок питания для микроконтроллера
- Кнопки тактильные
- Светодиоды трехцветные
- Провода
- Корпусные детали для монтажа электроники



Теоретический материал

В современном мире электронные устройства становятся все более сложными и универсальными, объединяя в себе множество функций и возможностей. Одной из ключевых задач разработки таких устройств является организация эффективного взаимодействия пользователя с устройством. Для этой цели часто используются тактильные кнопки, LCD-дисплеи и светодиоды.

Тактильные кнопки (механические переключатели) служат для ввода данных пользователем. Кнопки устроены таким образом, что при нажатии замыкаются контакты, что генерирует электрический сигнал. Этот сигнал затем поступает в микроконтроллер или другую управляющую часть устройства для дальнейшей обработки.

Принцип действия тактильных кнопок:

1. Нажатие кнопки замыкает контакты и образует электрическую цепь.
2. Сигнал от кнопки поступает на вход микроконтроллера.
3. Микроконтроллер регистрирует нажатие и выполняет соответствующее программное действие.

LCD-дисплей (Liquid Crystal Display, жидкокристаллический дисплей) используется для отображения информации, такой как текст, изображения или графики. Он состоит из матрицы пикселей, каждый из которых может изменять свои оптические свойства под действием электрического сигнала, отображая таким образом нужную информацию.

Этапы отображения информации на LCD-дисплее:

1. Микроконтроллер передает данные на дисплей.
2. Данные преобразуются в электрические сигналы, которые управляют состоянием каждого пикселя.
3. Пиксели изменяют свои свойства, создавая изображение, видимое пользователю.

Светодиоды (LED, Light Emitting Diode) используются для индикации различных состояний устройства. Эти компоненты сверхяркие, экономичные и имеют долгий срок службы. Светодиоды управляются сигналами от микроконтроллера, который регулирует их включение и выключение.

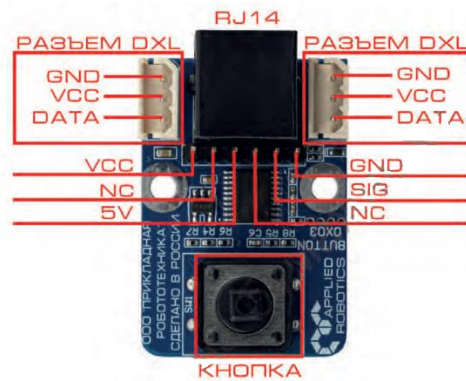
Работа светодиодной индикации:

4. Микроконтроллер посылает сигнал на включение/выключение светодиода.
5. Сигнал приводит к тому, что через светодиод начинает течь электрический ток.
6. Светодиод излучает свет, индицируя определенное состояние устройства.

Эффективная работа всех описанных компонентов предполагает их взаимодействие посредством микроконтроллера.

Таким образом, принцип работы взаимодействия тактильных кнопок, дисплея и индикации светодиодами заключается в комплексной обработке и отображении информации, что позволяет пользователю эффективно взаимодействовать с устройством. Центральным элементом в этом процессе является микроконтроллер, который управляет всеми компонентами и обеспечивает их слаженную работу в соответствии с логикой работы устройства. Модули ввода: модуль переключателя - тактовая кнопка, модуль потенциометра, модуль концевого микропереключателя, модуль ИК приемника.

Модуль «Тактовая кнопка» является обычной тактовой кнопкой, размещенной на плате с микроконтроллером. Данный модуль предполагается использовать для более удобного подключения кнопок к популярным микроконтроллерам. Внешний вид модуля представлен на рисунке.



Модуль «Тактовая кнопка» имеет:

- Разъем DXL - два трехпиновых разъема типа Molex, содержащих в себе линии GND (земля), VCC (питание), DATA (линия данных). Используются для подключения модуля по интерфейсу Dynamixel, как в одиночном виде, так и в составе цепи устройств.
- Разъем типа RJ14 для подключения модуля в фирменную плату расширения для подключения сенсорных модулей.
- Шестипиновый разъем.

Обычно тактильные кнопки используются для ввода команд или данных (например, выбор опции или ввод символов).

Дисплей подключается к микроконтроллеру Arduino через специальный драйвер (например, HD44780). Для работы с дисплеем используются библиотеки, например, LiquidCrystal, которые обеспечивают удобный интерфейс для отображения текста, символов и т.д. Для вывода информации на дисплей используются функции, например lcd.print().

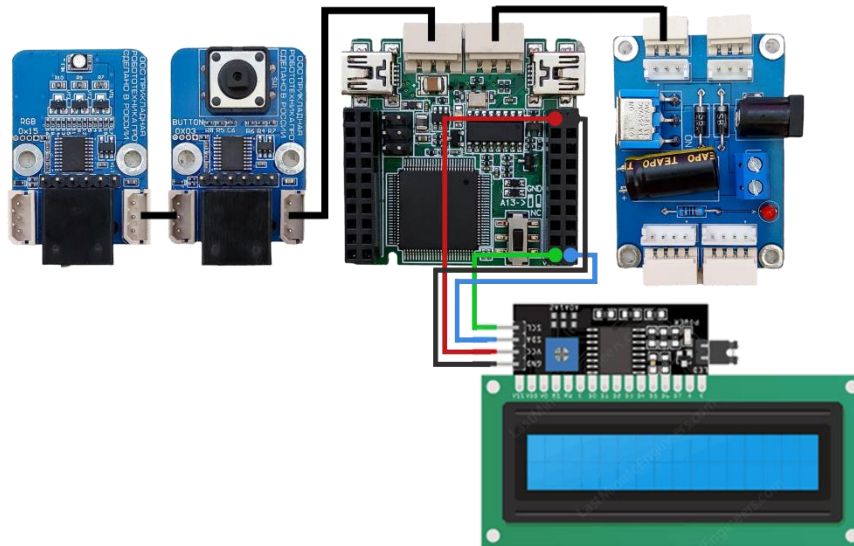
Практическая часть

Задания:

1. Собрать корпус удаленного терминала, используя детали и необходимую электронику из набора «Стем мастерская»



2. Подключить компоненты (светодиоды, кнопки, дисплей) к микроконтроллеру. Для проверки работоспособности датчиков можно воспользоваться скетчем Console_v2 из примера библиотеки DxlMaster2. С помощью данного скетча можно изменять ID.



3. Написать программный код для обработки ввода с кнопок и вывода на дисплей (за основу можно взять фрагмент кода из приложения).

Описание регистра модуля «Тактовая кнопка»

ID	Наименование регистра	Адрес	Тип данных	Допустимый диапазон
0x03	BUT_DATA	27	uint8_t	0..1

4. Реализовать алгоритм контроля состояния светодиодов в соответствии с данными ввода.
5. Провести тестирование работы терминала.

Практические примеры подключения модулей к контроллерам и примеры программ описаны в учебном пособии «Периферийные функциональные модули. Часть 1» (ООО «Прикладная робототехника»).

Контрольные вопросы

1. Какие компоненты необходимы для создания интерактивной системы терминала?
2. Какую функцию выполняют светодиоды в данной системе и каков механизм их управления?
3. Какие этапы нужно пройти для успешной отладки программного кода в подобных проектах?

Фрагмент кода

```
#include "DxlMaster.h"
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 20, 2);
String LCDLine = "";
String LCDLine1 = "";
uint8_t lineNumber = 0;
const unsigned long dynamixel_baudrate = 57600;
const unsigned long serial_baudrate = 57600;
const uint8_t id1 = 1;
const uint8_t idB1 = 1;
uint8_t b1;

DynamixelDevice light1(id1);
DynamixelDevice button1(idB1);

void setup() {
  DxlMaster.begin(dynamixel_baudrate);
  light1.init();
  button1.init();
  Serial.begin(serial_baudrate);
  lcd.init();
  lcd.backlight();
}

void loop() {
  button1.read(27, b1);
  if (b1 == 1){
    TL(light1, 26, 255, 0);
    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("Кнопка №1");
  }
}

void TL (DynamixelDevice light, int REG, int VAL1, int VAL2)
{
  light.write(REG, VAL1);
  delay(500);
  light.write(REG, VAL2);
  delay(1000);
}
```

Цель: изучить принципы управления сервоприводом с помощью микроконтроллера, разработать программу для управления позицией сервопривода и отображения информации о его состоянии.

Планируемые результаты:

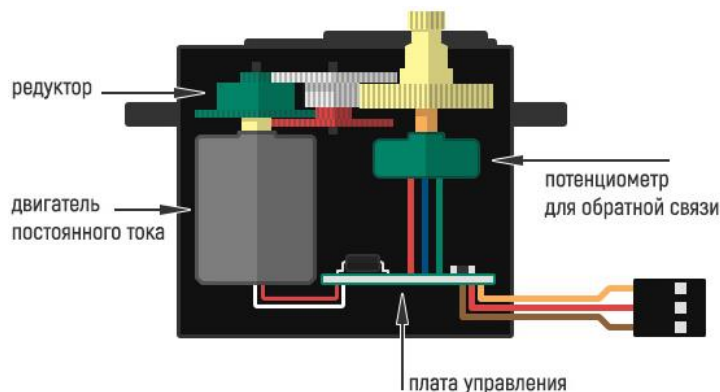
- знать структуру программы управления сервоприводом
- уметь программировать микроконтроллер для движения сервопривода

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер DXL-IOT
- Микроконтроллер OpenCM
- Блок питания микроконтроллера OpenCM и DXL-IOT
- Сервопривод Dynamixel
- Провода для подключения
- Преобразователь питания DXL-PWR для микроконтроллера DXL-IOT
- USB DXL AR



Теоретический материал



Сервопривод - это устройство, предназначенное для точного управления углом поворота или положением детали. Основными компонентами сервопривода являются двигатель, обратная связь и контроллер.

Сервоприводы широко используются в приложениях, требующих точного позиционирования, таких как робототехника, автоматизация процессов, управление камерами и многие другие. Основной компонент, обеспечивающий корректную работу сервопривода, — это контроллер. Контроллер в сервоприводе отвечает за управление двигателем на основе информации обратной связи и команд, поступающих от внешнего управляющего устройства.

Принципы работы контроллера в сервоприводе.

Контроллер в сервоприводе выполняет несколько ключевых функций:

1. Получение команд от внешнего управляющего устройства: Контроллер получает команды управления от внешнего устройства, например, микроконтроллера (Arduino, Raspberry Pi и т. д.). Эти команды обычно содержат информацию о целевой позиции, скорости или усилии, которые должен достичь сервопривод.
2. Считывание обратной связи: В современных сервоприводах используются сенсоры, такие как энкодеры или потенциометры, для получения информации о текущем положении, скорости и других параметрах двигателя. Сигналы от этих сенсоров поступают на вход контроллера.
3. Регулирование подачи электропитания на двигатель: На основе информации об обратной связи и команд управления контроллер регулирует подачу электропитания на двигатель. Это позволяет отклонять вал двигателя до нужного положения с заданной точностью и скоростью.

Для выполнения своих задач контроллеры сервоприводов используют различные алгоритмы управления. Наиболее распространенные из них:

Пропорционально-интегрально-дифференциальное (PID) управление:

1. Пропорциональная (P) составляющая корректирует сигнал управления пропорционально ошибке (разнице между текущим и целевым положением).
2. Интегральная (I) составляющая учитывает накопленную ошибку за время, корректируя сигнал управления, чтобы устранить постоянные систематические ошибки.
3. Дифференциальная (D) составляющая учитывает скорость изменения ошибки, что позволяет предотвратить перерегулирование и слишком быстрые изменения в управлении.

Простейшее управление типа «все или ничего»:

Контроллер включает и выключает подачу энергии на двигатель в ответ на определенные условия. Это простой, но менее точный метод управления.

Продвинутая адаптивная система управления:

Использует сложные математические модели для адаптивного управления, автоматически настраивая параметры управления в зависимости от изменения динамики системы и различных условий работы.

Рассмотрим пример с использованием контроллера в сервоприводе, управляемом Arduino:

1. Инициализация: Сначала микроконтроллер Arduino отправляет начальную команду на контроллер сервопривода, указывая начальную позицию вала.
2. Команда на изменение положения: Пользователь через интерфейс отправляет команду, изменяющую целевую позицию. Arduino передает эту команду контроллеру сервопривода.
3. Обработка команды: Сервоконтроллер получает новую команду и считывает текущее положение вала с энкодера.
4. Оценка ошибки: Контроллер вычисляет ошибку (разницу) между текущей и целевой позициями.
5. Применение управленческого сигнала: На основе алгоритма управления (например, PID) контроллер генерирует сигнал, регулирующий подачу электропитания на двигатель, корректирует скорость и направление вращения вала.
6. Достижение цели: Когда ошибка минимальна или равна нулю (целевое положение достигнуто), контроллер снижает или прекращает подачу энергии, удерживая вал в заданной позиции.

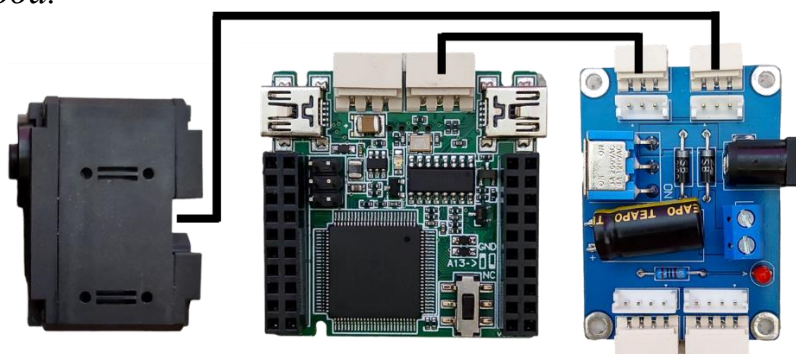
Контроллер в сервоприводе является критически важным компонентом, обеспечивающим точное и надежное управление двигателем. Он отвечает за обработку внешних команд, получение данных об обратной связи и регулицию подачи энергии на двигатель. Применяемые алгоритмы управления, такие как PID, позволяют добиваться высокой точности и стабильности в работе сервопривода, что существенно расширяет его применение в различных областях техники и автоматизации.

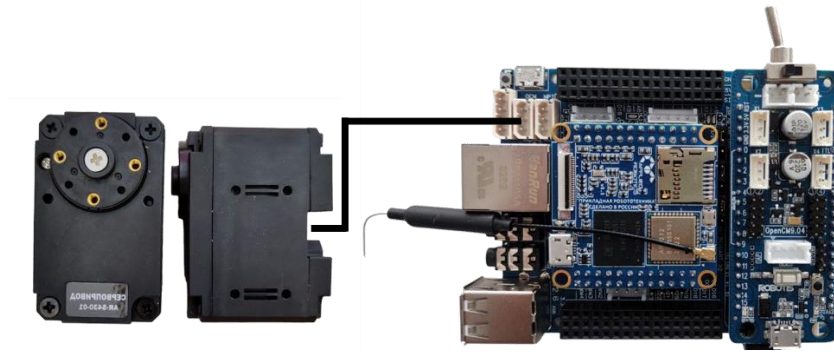
Практическая работа

Задания:

1. Подключить сервоприводы к микроконтроллеру.

Для проверки работоспособности, исправности, а также для настройки сервопривода можно воспользоваться утилитой Dynamixel Wizard. С помощью данного приложения удобно изменять стандартные настройки сервопривода, такие как: ID, Baudrate и т.д. Также утилита позволяет обновить прошивку сервопривода.





2. Написать программу управления с использованием библиотеки Dynamixel2Arduino для плат DXL-IOT и OpenCM (за основу можно взять фрагмент кода из приложения)
3. Запрограммировать функции установки позиции, чтения текущей позиции и вывода в монитор порта Arduino IDE.
4. Загрузить программу на Arduino, подключить к компьютеру и тестировать управление.

Контрольные вопросы

1. Что такое сервопривод и для чего он используется?
2. Какие параметры влияют на точность позиционирования сервопривода?
3. Какая функция Arduino используется для управления сервоприводом?

Фрагмент кода для платы STEM с OpenCM 9.04

```
#include <Dynamixel2Arduino.h>
#define DXL_SERIAL Serial3
#define DEBUG_SERIAL Serial
const uint8_t DXL_DIR_PIN = 22;
const float DXL_PROTOCOL_VERSION = 2.0;
Dynamixel2Arduino dxl(DXL_SERIAL, DXL_DIR_PIN);
int joint=1;

void setup() {
  DEBUG_SERIAL.begin(57600);
  dxl.begin(1000000);
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
  dxl.torqueOff(joint);
  dxl.setOperatingMode(joint, OP_POSITION);
}

void loop() {
  dxl.torqueOn(joint);
  dxl.setGoalVelocity(joint, 5);
  dxl.setGoalPosition(joint, 4000);
  delay(100);
  dxl.setGoalVelocity(joint, 5);
  dxl.setGoalPosition(joint, 8000);
  delay(100);
}
```

Фрагмент кода для платы DXL-IOT

```
#include "DxlMaster2.h"
const uint8_t id = 1;
int16_t speed = 200;
const long unsigned int baudrate = 57600;
uint8_t led_state = true;
DynamixelMotor motor(id, 1);

void setup()
{
  DxlMaster.begin(baudrate);
  delay(100);
  uint8_t buf[3];
  uint8_t status;
  status = motor.ping(buf);
  uint16_t num = buf[0] | (buf[1] << 8);
  motor.enableTorque(0);
  motor.write(DYN2_ADDR_OPERATION_MODE, (uint8_t)1);
  motor.enableTorque(1);
}

void loop()
{
  motor.speed(speed);
  speed = -speed;
  motor.write(DYN2_ADDR_LED, (uint8_t)led_state);
  led_state = !led_state;
  delay(3000);
}
```

Цель: изучить принципы программирования управления манипулятором с угловой кинематикой для перемещения объекта.

Предметные результаты урока:

- знать структуру программы управления манипулятором
- уметь запрограммировать микроконтроллер для перемещения манипулятором с угловой кинематикой объекта из точки А в точку Б

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер OpenCM
- Блок питания микроконтроллера OpenCM
- Сервоприводы Dynamixel
- Провода для подключения
- Детали для сборки манипулятора с угловой кинематикой
- Объекты для перемещения

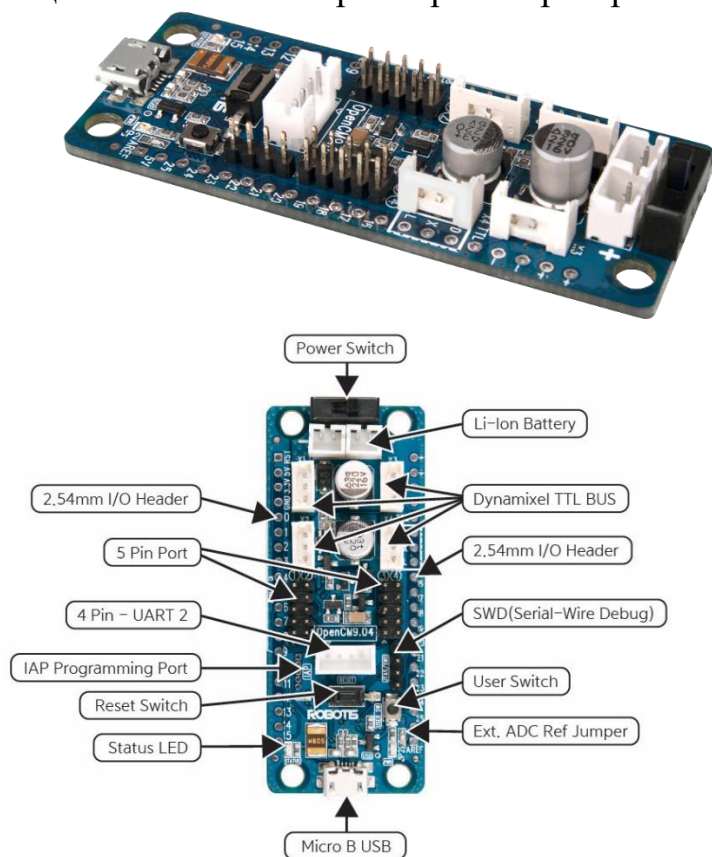


Теоретический материал

Угловая кинематика является одним из важных аспектов в управлении робототехническими манипуляторами. Она определяет взаимосвязь между углами поворота суставов (звеньев) манипулятора и его конечным положением в пространстве. Для манипулятора с угловой кинематикой задача состоит в том, чтобы с помощью серии углов поворота каждого сустава достичь точного положения объекта в трехмерном пространстве.

При программировании манипулятора с угловой кинематикой необходимо учитывать кинематические параметры каждого звена, такие как длины и углы поворота, чтобы точно определить конечное положение манипулятора. Эти параметры помогают рассчитать траекторию движения каждого звена и вычислить углы поворота, необходимые для достижения заданной точки.

Угловая кинематика позволяет манипулятору ориентироваться в пространстве и перемещать объекты с высокой точностью и эффективностью. Понимание принципов угловой кинематики помогает инженерам и программистам разрабатывать сложные управляющие системы для роботов и автоматизированных механизмов, обеспечивая точное и плавное перемещение объектов в трехмерном пространстве.



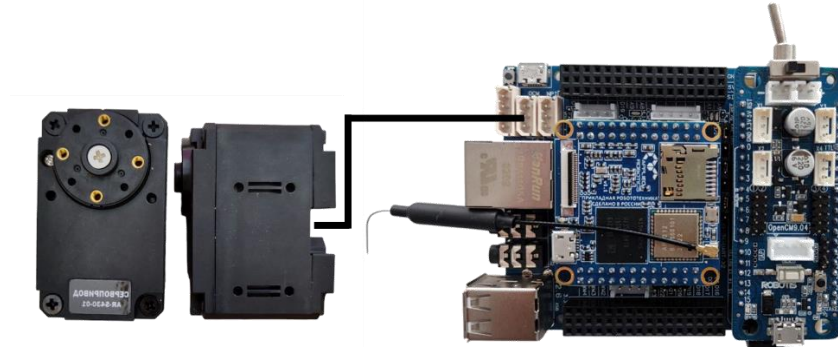
Программируемый контроллер OpenCM9.04, представляет собой контроллер с открытым исходным кодом, ориентирован на применение совместно с компонентами и программными продуктами, предлагаемыми компанией ROBOTIS. По своей структуре контроллер OpenCM9.04 является Arduino-подобной платой, но предоставляет более широкие возможности по сравнению со стандартной платформой Arduino. Контроллер OpenCM9.04 позволяет подключать к себе не только фирменные компоненты, но также и компоненты сторонних производителей - т.е., может быть использован как плата сбора данных. В то же время набор стандартных интерфейсов контроллера может быть применен для управления и взаимодействия с различными внешними устройствами, например, модулями беспроводной связи, сложными комплексными устройствами и т.д.

Практическая работа

Задания:

1. Подключить сервоприводы к микроконтроллеру

Для проверки работоспособности, исправности, а также для настройки сервопривода можно воспользоваться утилитой Dynamixel Wizard 2.0. С помощью данного приложения удобно изменять стандартные настройки сервопривода, такие как: ID, Vaudrate и т.д. Также утилита позволяет обновить прошивку сервопривода.



2. Задать начальные координаты точки А и конечные координаты точки Б для перемещения объекта.

Можно воспользоваться ячейками, напечатанными на 3D-принтере ([скачать файл для печати](#)).



3. Написать программу на языке программирования для управления манипулятором (за основу можно взять фрагмент кода из приложения)

4. Подключить манипулятор к компьютеру и загрузите программу.

5. Запустить программу и провести тестирование перемещения объекта между точками.

Контрольные вопросы

1. Какое количество серводвигателей используется в манипуляторе с угловой кинематикой?
2. Какие параметры описывают положение объекта в пространстве при использовании угловой кинематики?
3. Какие алгоритмы можно использовать для планирования траектории движения манипулятора?

Фрагмент кода (вращение сервоприводов)

```
#include <Dynamixel2Arduino.h>
#define DXL_SERIAL Serial3
#define DEBUG_SERIAL Serial
#define DEGREE_COEFF 16383/360
const uint8_t DXL_DIR_PIN = 22;
const float DXL_PROTOCOL_VERSION = 2.0;
Dynamixel2Arduino dxl(DXL_SERIAL, DXL_DIR_PIN);

void setup() {
  DEBUG_SERIAL.begin(57600);
  dxl.begin(1000000);
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
  dxl.torqueOff(1);
  dxl.torqueOff(2);
  dxl.setOperatingMode(1, OP_POSITION);
  dxl.setOperatingMode(2, OP_POSITION);
  dxl.torqueOn(1);
  dxl.torqueOn(2);
}

void loop() {
  dxl.setGoalVelocity(1, 1);
  dxl.setGoalPosition(1, 180*DEGREE_COEFF);
  delay(5000);
  dxl.setGoalVelocity(2, 1);
  dxl.setGoalPosition(2, 135*DEGREE_COEFF);
  delay(5000);
}
```

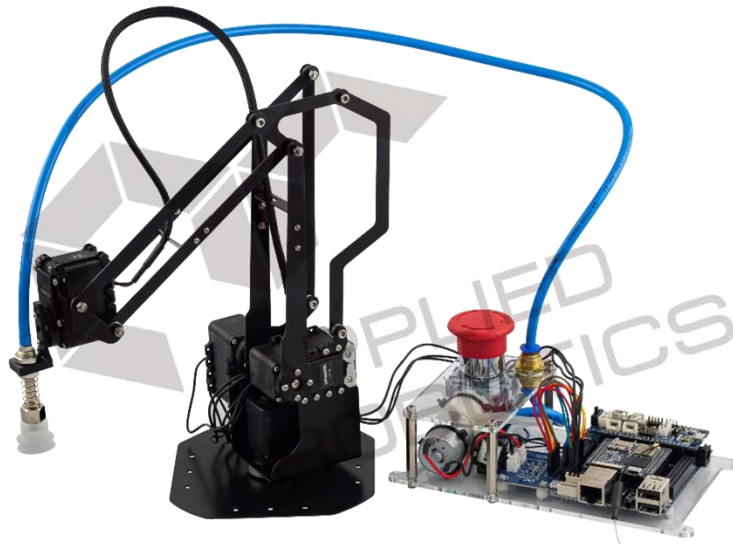
Цель: изучить принципы программирования управления манипулятором с плоскопараллельной кинематикой для перемещения объекта.

Планируемые результаты:

- знать структуру программы управления манипулятором
- уметь запрограммировать микроконтроллер для перемещения манипулятором с плоскопараллельной кинематикой объекта из точки А в точку Б

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер OpenCM
- Блок питания микроконтроллера OpenCM
- Сервоприводы Dynamixel
- Провода для подключения
- Детали для сборки манипулятора с плоскопараллельной кинематикой
- Объекты для перемещения



Теоретический материал

Манипуляторы с плоскопараллельной кинематикой характеризуются тем, что плоскости движения каждого акта, а также плоскости всех остальных актов планарны и параллельны друг другу. Это позволяет управлять манипулятором в двумерной плоскости с использованием собственных координат для точного позиционирования объекта.

Программирование манипулятора с плоскопараллельной кинематикой включает в себя задание необходимых углов и перемещений для каждого привода или звена. Это помогает определить точное положение объекта в пространстве с учетом плоскости перемещения.

Плоскопараллельная кинематика обеспечивает высокую точность перемещения объектов в двухмерном пространстве, что полезно при выполнении задач с небольшим количеством степеней свободы.

Основными преимуществами таких манипуляторов являются следующие особенности:

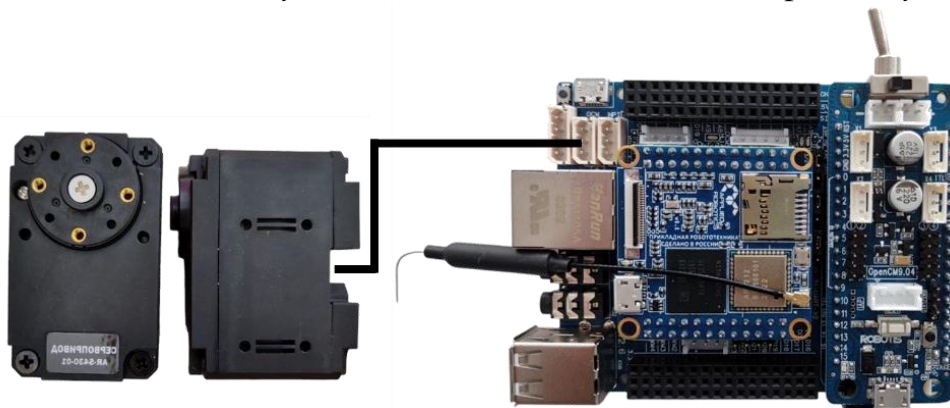
- **Плоскопараллельность.** Кинематика манипулятора позволяет ему перемещать объект в пределах плоскости, что позволяет упростить расчеты и программирование движений.
- **Параллельность плоскостей движения.** Каждый из приводов манипулятора управляется отдельно, что позволяет достичь параллельности плоскостей движения и ограничиться двумя степенями свободы.
- **Высокая точность.** Благодаря особенностям кинематической структуры, манипуляторы с плоскопараллельной кинематикой могут обеспечивать высокую точность перемещения объектов, что делает их особенно полезными в задачах, где требуется точное позиционирование.
- **Высокая скорость.** Управление приводами манипулятора позволяет достигать высоких скоростей перемещения объектов, что делает их эффективными в задачах, где требуется быстрое выполнение операций.

Практическая работа

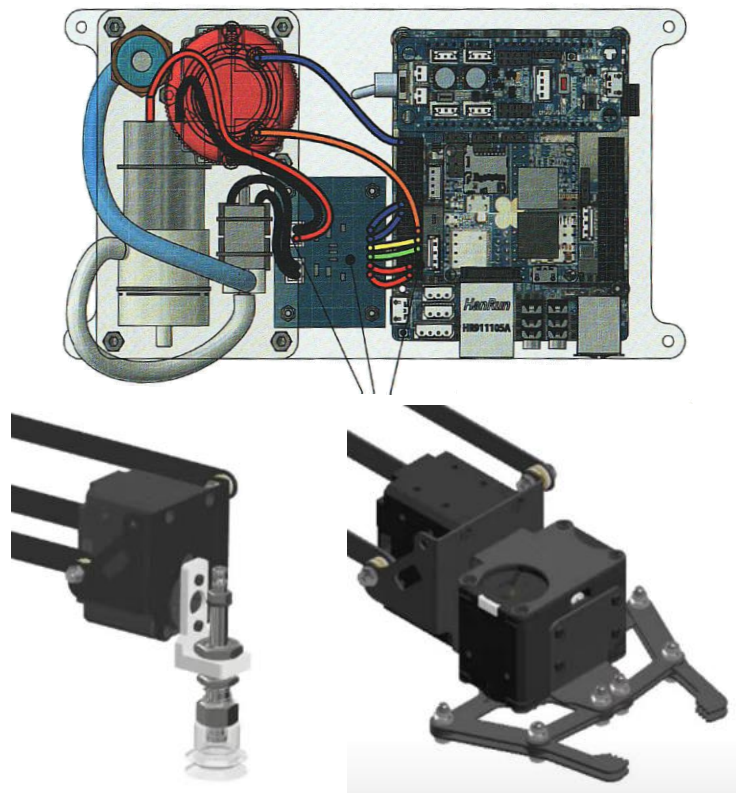
Задания:

1. Подключить сервоприводы к микроконтроллеру

Для проверки работоспособности, исправности, а также для настройки сервопривода можно воспользоваться утилитой Dynamixel Wizard 2.0. С помощью данного приложения удобно изменять стандартные настройки сервопривода, такие как: ID, Baudrate и т.д. Также утилита позволяет обновить прошивку сервопривода.



Возможно использование вакуумной присоски с пневмосистемой в качестве схвата.



2. Задать начальные координаты точки А и конечные координаты точки Б для перемещения объекта.

Можно воспользоваться ячейками, напечатанными на 3D-принтере ([скачать файл для печати](#)).



3. Написать программу на языке программирования для управления манипулятором (за основу можно взять фрагмент кода из приложения)
4. Подключить манипулятор к компьютеру и загрузите программу.
5. Запустить программу и провести тестирование перемещения объекта между точками.

Контрольные вопросы

1. Что такое плоскопараллельная кинематика манипулятора?
2. Какие преимущества обеспечивает использование манипулятора с плоскопараллельной кинематикой?
3. Какие этапы необходимо выполнить при программировании управления манипулятором с этой кинематикой?

Фрагмент кода

```

#include <Dynamixel2Arduino.h>
#define DXL_SERIAL Serial3
#define DEBUG_SERIAL Serial
const uint8_t DXL_DIR_PIN = 22;
const float DXL_PROTOCOL_VERSION = 2.0;
#define vacuum_key 15
#define vacuum_pump 14
Dynamixel2Arduino dxl(DXL_SERIAL, DXL_DIR_PIN);
using namespace ControlTableItem;
float pi = 3.14159265;
float rad2ticks = 2*pi / 16383;
float degree_coeff = 16383.0 / 360;
#define jointN 4

bool input_coords = true;
int start_pos[jointN+1];
int i = 0;
int alphas[jointN+1];
int V;
float X, Y, Z, x, y, z, xf, yf, zf, d, betta_d, gamma_d, alpha1, alpha2, alpha3,
alpha4, alpha5, l1 = 0.05, l2 = 0.14, l3 = 0.14, l4 = 0.07, l5 = 0.02;

void setup() {
  pinMode(vacuum_pump, OUTPUT);
  pinMode(vacuum_key, OUTPUT);
  digitalWrite(vacuum_pump, LOW);
  digitalWrite(vacuum_key, LOW);
  DEBUG_SERIAL.begin(57600);
  dxl.begin(1000000);
  dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
  for (i=1; i<=jointN; i++)
  {
    dxl.torqueOff(i);
    dxl.setOperatingMode(i, OP_POSITION);
  }
  delay(5000);
  for (i=1; i<=jointN; i++)
  {
    start_pos[i] = 8192;//int(180 * degree_coeff);
    dxl.torqueOn(i);
    dxl.writeControlTableItem(PROFILE_VELOCITY, i, 50);
    dxl.setGoalPosition(i, start_pos[i]);
  }
  delay(2000);
}

void loop() {
  if (input_coords) {
    DEBUG_SERIAL.println("Введите координаты цели x,y,z в сантиметрах через пробел:");
    input_coords = false;
  }
  if (DEBUG_SERIAL.available()) {
    // X = 0.15;
    // Y = 0.0;
    // Z = 0.07;
    parse_input(DEBUG_SERIAL.readString());
    if (V == 1)
    {
      digitalWrite(vacuum_pump, HIGH);
      digitalWrite(vacuum_key, HIGH);
    }
    else
    {
      digitalWrite(vacuum_pump, LOW);
      digitalWrite(vacuum_key, LOW);
    }
    alpha1 = atan(Y / X);
  }
}

```

```

z = z - 11 + 15;
x = max(14, x - 14);
DEBUG_SERIAL.print("x = "); DEBUG_SERIAL.println(x);
DEBUG_SERIAL.print("y = "); DEBUG_SERIAL.println(y);
DEBUG_SERIAL.print("z = "); DEBUG_SERIAL.println(z);
DEBUG_SERIAL.print("alpha1 = "); DEBUG_SERIAL.println(alpha1);
d = sqrt(x*x + z*z);

gamma_d = acos((12*12 + d*d - 13*13)/(2*12*d));
beta_d = atan(z / x);
alpha2 = pi/2 - gamma_d - beta_d;
alpha3 = pi - acos((12*12 + 13*13 - d*d)/(2*12*13));
alpha3 = alpha3 - (pi/2 - alpha2);
xf = cos(alpha1)*(12*sin(alpha2) + 13*sin(pi-alpha3-alpha2));
yf = sin(alpha1)*(12*sin(alpha2) + 13*sin(pi-alpha3-alpha2));
zf = 12*cos(alpha2) - 13*cos(pi-alpha3-alpha2);
DEBUG_SERIAL.print("xf = "); DEBUG_SERIAL.println(xf);
DEBUG_SERIAL.print("yf = "); DEBUG_SERIAL.println(yf);
DEBUG_SERIAL.print("zf = "); DEBUG_SERIAL.println(zf);
delay(1000);
alpha4 = 0;
alpha5 = 0;
alphas[1] = start_pos[1] + int(alpha1/rad2ticks);
alphas[2] = start_pos[2] + int(alpha2/rad2ticks);
alphas[3] = start_pos[3] + int(alpha3/rad2ticks);
alphas[4] = start_pos[4] + int(alpha4/rad2ticks);
alphas[5] = start_pos[5] + int(alpha5/rad2ticks);
for (i=1; i<=3; i++)
dx1.setGoalPosition(i, alphas[i]);
delay(1000);
input_coords = true;
}
}

void parse_input(String str) {
int separator = str.indexOf(' ');
X = str.substring(0, separator).toInt()/100.0;
int separator2 = str.indexOf(' ', separator+1);
Y = str.substring(separator+1, separator2).toInt()/100.0;
int separator3 = str.indexOf(' ', separator2+1);
Z = str.substring(separator2+1, separator3).toInt()/100.0;
V = str.substring(separator3).toInt();
}

```

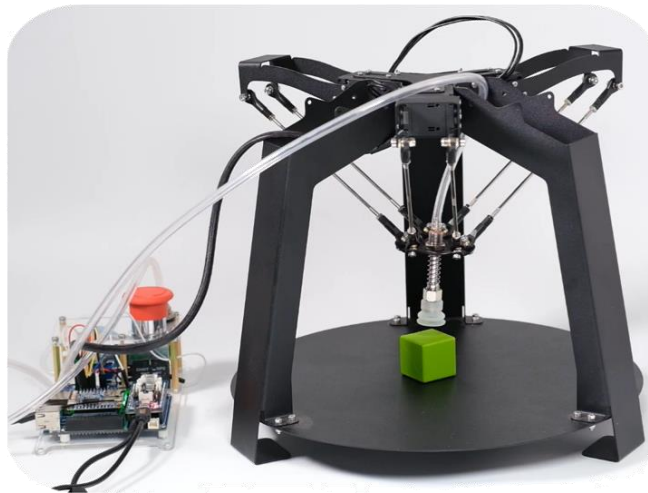
Цель: изучить принципы программирования управления манипулятором с дельта-кинематикой для перемещения объекта.

Планируемые результаты:

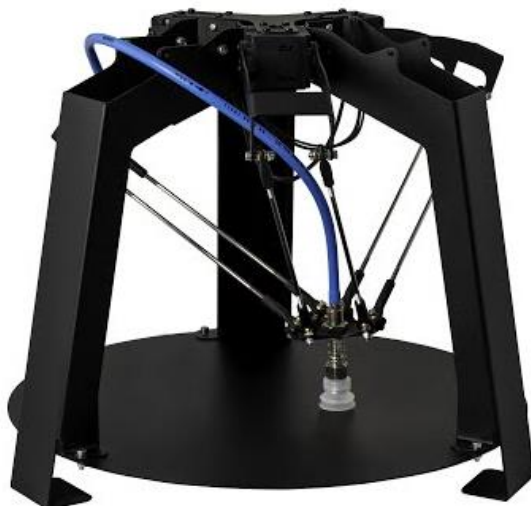
- знать структуру программы управления манипулятором
- уметь запрограммировать микроконтроллер для перемещения манипулятором с дельта-кинематикой объекта из точки А в точку Б

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер OpenCM
- Блок питания для микроконтроллера OpenCM
- Сервоприводы Dynamixel
- Провода для подключения
- Детали для сборки манипулятора с дельта-кинематикой
- Объекты для перемещения



Теоретический материал



Манипуляторы с дельта-кинематикой представляют собой систему из трех параллельно расположенных манипуляторов, которые работают с координатами в пространстве. Они обладают специфической кинематикой, которая позволяет им обеспечивать высокую скорость перемещения и точность при выполнении задач позиционирования объектов в трехмерном пространстве.

Основные компоненты манипулятора с дельта-кинематикой обычно включают в себя эффектор (рабочий орган) и приводы (обычно шаговые двигатели) для управления каждым из трех параллельных манипуляторов. Система координат такого манипулятора определяется в соответствии с его конструкцией и расположением.

Применение дельта-кинематики позволяет достичь следующих преимуществ:

- Высокая скорость и динамические характеристики. Параллельная конфигурация манипулятора позволяет достигнуть высокой скорости перемещения рабочего органа, что особенно важно в операциях, требующих быстроты.
- Высокая точность позиционирования. Благодаря особенностям кинематики и механизма работы манипулятора с дельта-кинематикой можно добиться высокой точности позиционирования объектов в трехмерном пространстве.
- Отличная параллельная ориентация. Манипуляторы с дельта-кинематикой хорошо подходят для операций, где необходимо управление в нескольких координатах одновременно, что обеспечивает эффективность в работе с объектами в трехмерном пространстве.

Таким образом, манипуляторы с дельта-кинематикой представляют собой эффективное решение для выполнения задач быстрого и точного позиционирования объектов в трехмерном пространстве, таких как сборка, печать 3D-моделей, обслуживание оборудования и другие приложения, где требуется высокая динамичность и точность работы.

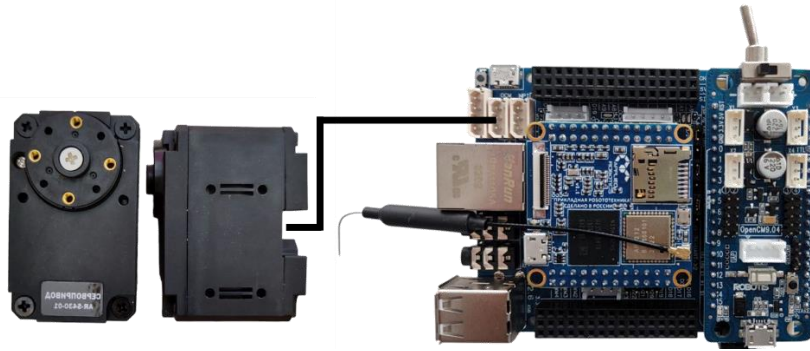
Практическая работа

Задания:

1. Подключить сервоприводы к микроконтроллеру

*Для проверки работоспособности, исправности, а также для настройки сервопривода можно воспользоваться утилитой *Dynamixel Wizard 2.0*. С помощью данного приложения удобно изменять стандартные настройки сервопривода,*

такие как: ID, Baudrate и т.д. Также утилита позволяет обновить прошивку сервопривода.



2. Задать начальные координаты точки А и конечные координаты точки Б для перемещения объекта.

Можно воспользоваться ячейками, напечатанными на 3D-принтере ([скачать файл для печати](#)).



3. Написать программу на языке программирования для управления манипулятором (за основу можно взять фрагмент кода из приложения)
4. Подключить манипулятор к компьютеру и загрузите программу.
5. Запустить программу и провести тестирование перемещения объекта между точками.

Контрольные вопросы

1. Что представляет собой манипулятор с дельта-кинематикой? Каковы его основные преимущества?
2. Какие этапы необходимо выполнить для успешной реализации программы управления манипулятором?
3. Как можно оценить эффективность и точность работы манипулятора с дельта-кинематикой? Какие факторы могут повлиять на точность позиционирования объектов?

Фрагмент кода

```

#include <DynamixelWorkbench.h>
#ifdef __OPENC904__
#define DEVICE_NAME "3"
#elif defined(__OPENC9__ )
#define DEVICE_NAME ""
#endif
#define BAUDRATE 1000000
#define DXL_ID1 1
#define DXL_ID2 2
#define DXL_ID3 3
#define vacuum_key 11
#define vacuum_pump 12
DynamixelWorkbench dxl_wb;
int f = 60;
int Rf = 90;
int Re = 168;
int e = 52;
float theta1, theta2, theta3, theta;
float y_1, y_2, y_3;
float x_1, x_2, x_3;
float m, n, k, v;

void coord(float a, float b)
{
    x_1 = a;
    y_1 = b;
    x_2 = a*cos(2.094) + b*sin(2.094);
    y_2 = b*cos(2.094) - a*sin(2.094);
    x_3 = a*cos(2.094) - b*sin(2.094);
    y_3 = b*cos(2.094) + a*sin(2.094);
}

void find_angle(float x, float y, float z)
{
    float c1 = -0.5 * 0.57735 * f;
    y -= 0.5 * 0.57735 * e;
    float a = (x*x + y*y + z*z + Rf*Rf - Re*Re - c1*c1)/(2*z);
    float b = (c1-y)/z;
    float d = -(a+b*c1)*(a+b*c1)+Rf*(b*b*Rf+Rf);
    if(d < 0)
        Serial.println("Error");
    float yj = (c1 - a*b - sqrt(d))/(b*b + 1);
    float zj = a + b*yj;
    float angle = atan(-zj/(c1 - yj));
    theta = angle;
}

void full(float x0, float y0, float z0)
{
    coord(x0, y0);
    find_angle(x_1, y_1, z0);
    theta1 = theta;
    find_angle(x_2, y_2, z0);
    theta2 = theta;
    find_angle(x_3, y_3, z0);
    theta3 = theta;
}

void set_pos( float p1 , float p2, float p3, int de1)
{
    dxl_wb.goalPosition(DXL_ID1,p1);
    dxl_wb.goalPosition(DXL_ID2,p2);
    dxl_wb.goalPosition(DXL_ID3,p3);
    delay(de1);
}

void setup()
{

```

```

    pinMode(vacuum_pump, OUTPUT);
    pinMode(vacuum_key, OUTPUT);
const char *log ;
Serial.begin(9600);
uint16_t model_number1 = 1;
uint16_t model_number2 = 2;
uint16_t model_number3 = 3;
dxl_wb.init(DEVICE_NAME, BAUDRATE, &log);
dxl_wb.ping(DXL_ID1, &model_number1, &log);
dxl_wb.ping(DXL_ID2, &model_number2, &log);
dxl_wb.ping(DXL_ID3, &model_number3, &log);
dxl_wb.jointMode(DXL_ID1, 50, 0, &log);
dxl_wb.jointMode(DXL_ID2, 50, 0, &log);
dxl_wb.jointMode(DXL_ID3, 50, 0, &log);
}

void loop() {
while (Serial.available() > 0 && Serial.read() == 'c') {
    m = Serial.parseInt();
    n = Serial.parseInt();
    k = Serial.parseInt();
    v = Serial.parseInt();
    Serial.print("X = " );
    Serial.println(m);
    Serial.print("Y = " );
    Serial.println(n);
    Serial.print("Z = " );
    Serial.println(k);
    if (k > 0)
    {
        k = 0;
        Serial.println("Z only negativ number");
    }
    if (v == 1){
        Serial.println("vacuum On" );
        digitalWrite(vacuum_pump, HIGH);
        digitalWrite(vacuum_key, HIGH);
    }
    else{
        Serial.println("vacuum Off" );
        digitalWrite(vacuum_pump, LOW);
        digitalWrite(vacuum_key, LOW);
    }
    full(m, n, k);
    Serial.print("Angle #1: " );
    Serial.println(theta1);
    Serial.print("Angle #2: " );
    Serial.println(theta2);
    Serial.print("Angle #3: " );
    Serial.println(theta3);
    set_pos(theta1, theta2, theta3, 500);
}
}
}

```

Цель: изучить принципы программирования управления манипулятором для перемещения объекта другому манипулятору.

Планируемые результаты:

- знать структуру программы управления манипуляторами
- уметь программировать микроконтроллер для перемещения манипулятором объекта другому манипулятору

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер OpenCM
- Блок питания микроконтроллера OpenCM
- Сервоприводы Dynamixel
- Провода для подключения
- Детали для сборки манипуляторов
- Объекты для перемещения



Теоретический материал

Взаимодействие двух роботов-манипуляторов — это интересная и важная тема в области робототехники и автоматизации. Такие системы часто используются в различных промышленных и научных приложениях, включая сборку, упаковку, медицинское обслуживание и многие другие сферы.

Основные аспекты взаимодействия роботов-манипуляторов.

Синхронизация движений:

Взаимодействие двух роботов часто требует точной синхронизации их движений. Это может быть достигнуто с помощью алгоритмов управления, которые обеспечивают координацию действий. Например, если один манипулятор поднимает объект, другой должен быть готов его поймать или переместить. Использование датчиков, таких как камеры или ЛИДАР, может помочь в отслеживании положения объектов и корректировке движений.

Обмен данными:

Роботы могут взаимодействовать друг с другом через систему обмена данными. Это может включать в себя использование проводных или беспроводных соединений (например, Wi-Fi или Bluetooth) для передачи информации о своем состоянии, положении и задачах. Взаимодействие в реальном времени является критически важным для успешного выполнения совместных задач.

Совместные задачи:

Взаимодействие роботов может быть организовано для выполнения совместных задач, например, когда один робот ведет за собой, а другой следует за ним, или когда один робот выполняет сборку, а другой — упаковку. Применение подходов, таких как распределенное управление и параллельная обработка, позволяет значительно увеличить эффективность производственных процессов.

Избежание коллизий:

Важно, чтобы во время взаимодействия роботов не происходило столкновений. Для этого используются алгоритмы, позволяющие заранее планировать путь каждого манипулятора с учетом возможного контакта. Это может включать динамическое измерение расстояний между манипуляторами и их окружением.

Адаптация и обучение:

Роботы могут обучаться взаимодействию друг с другом. Использование методов машинного обучения и нейронных сетей позволяет создать алгоритмы, которые адаптируются к изменениям в среде и оптимизируют свои действия на основе предыдущего опыта. Например, после нескольких операций робот может эффективно изучить лучшие способы передачи объектов друг другу.

Взаимодействие роботов-манипуляторов находит применение в таких сферах, как автоматизация производственных линий, где требуется высокая скорость и точность. В медицине роботизированные системы могут использоваться для хирургических операций, где два робота могут работать в тандеме, поддерживая друг друга для обеспечения лучших результатов.

Сборка автомобилей. На конвейерных линиях, где требуется высокая точность и скорость, использование пары роботов-манипуляторов позволяет одновременно выполнять различные операции, такие как сварка и установка деталей, значительно увеличивая производительность и снижая вероятность ошибок.

Логистика. В логистических центрах два робота могут взаимодействовать для перемещения товаров и упаковки. Один манипулятор может захватывать и перемещать

коробки, в то время как другой обеспечивает их правильное размещение на палетах или в контейнерах.

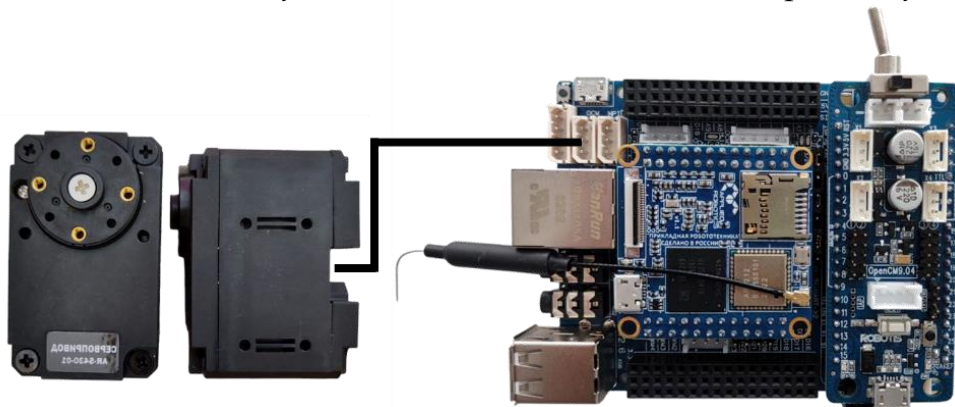
Научные исследования. В лабораториях два робота могут работать над экспериментами, где один манипулятор выполняет подготовку образцов, а другой — анализ. Это позволяет сократить время на отчеты и повысить эффективность работы исследовательских групп.

Взаимодействие роботов-манипуляторов открывает новые горизонты в автоматизации и производственных процессах. Развитие технологий, таких как искусственный интеллект и сенсорные системы, продолжает улучшать их взаимодействие, делая его более комфортным и эффективным. Возможности применения таких систем остаются практически безграничными, и с каждым годом мы можем наблюдать их все более широкое использование в различных отраслях.

Практическая работа

Задания:

1. Подключить сервоприводы к микроконтроллеру
Для проверки работоспособности, исправности, а также для настройки сервопривода можно воспользоваться утилитой Dynamixel Wizard 2.0. С помощью данного приложения удобно изменять стандартные настройки сервопривода, такие как: ID, Baudrate и т.д. Также утилита позволяет обновить прошивку сервопривода.



2. Задать начальные координаты точки А объекта перемещения для первого манипулятора и конечные координаты точки Б для второго манипулятора.
3. Написать программу на языке программирования для управления манипуляторами и перемещения объекта между друг другом.
4. Подключить манипуляторы к компьютеру и загрузите программу.
5. Запустить программу и провести тестирование перемещения объекта между точками.

Контрольные вопросы

1. Каковы основное назначение и функции роботов-манипуляторов в производстве?
2. В каких промышленных областях чаще всего применяют взаимодействие нескольких манипуляторов?
3. Какие основные проблемы могут возникнуть при взаимодействии двух роботов?

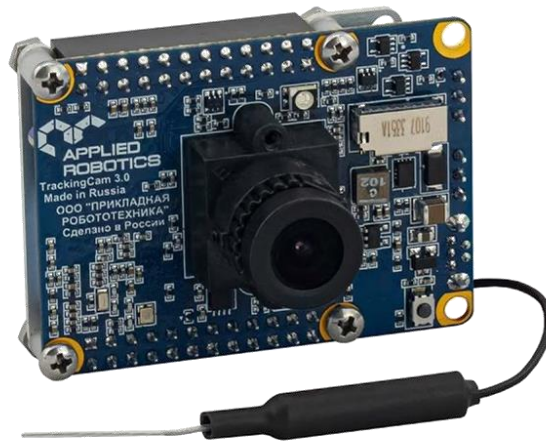
Цель: изучить принципы подключения и настройки модуля технического зрения для взаимодействия с манипулятором.

Предметные результаты:

- знать структуру программы управления манипулятором с помощью камеры технического зрения
- уметь настраивать и программировать камеру технического зрения

Используемое оборудование и материалы:

- Компьютер с Arduino IDE
- Микроконтроллер OpenCM и DXL-IOT
- Блоки питания для микроконтроллеров
- Сервоприводы Dynamixel
- Провода для подключения
- Модуль технического зрения TrackingCam3



Теоретический материал

Современные мобильные роботы функционируют в заранее неопределенной, изменяющейся среде, взаимодействуют с объектами в ней. В процессе движения робот должен оценивать обстановку: измерять собственные координаты, классифицировать окружающие его объекты и определять их положение. Для этого используются различные датчики и системы. Для определения координат робота служат спутниковая навигация, маяки различных принципов действия, колесная одометрия, инерциальные системы, лазерные дальномеры. Окружающие объекты также выделяют с помощью измеряющих различные физические величины датчиков: радиолокационных и ультразвуковых, тактильных, термометрических, химических, широкого круга оптических датчиков от фотореле до лазерных дальномеров и телекамер и др. Особое место среди оптических локационных систем занимают системы технического зрения (СТЗ).

Одной из ключевой задачи СТЗ является распознавание образов – заданных определенными признаками объектов сцены.

Процесс преобразования информации в СТЗ можно представить в виде шести основных этапов:

1. Ввод (восприятие) информации, т. е. получение изображения рабочей сцены с помощью датчиков;
2. Предварительная обработка изображения с использованием методов подавления шума и удаления искажений
3. Сегментация, т. е. выделение на изображении одного или нескольких представляющих интерес объектов сцены;
4. Описание, т. е. определение характерных параметров (цвета, размеров, формы и т. д.) каждого объекта, необходимых для его выделения на сцене;
5. Распознавание объекта, т.е. установление его принадлежности к некоторому классу деталей, например, к «кубикам»;
6. Интерпретация, т.е. получение искомых величин, в частности выявления принадлежности объекта к группе распознаваемых, например, «на сцене есть несколько кубиков».

Модуль TrackingCam3 представляет собой компактное устройство, оснащенное камерой и процессором компьютерного зрения, способное обрабатывать изображения в реальном времени. Благодаря специализированным алгоритмам распознавания объектов, TrackingCam3 может автоматически определять положение и характеристики объектов в кадре.

Основной функцией TrackingCam3 является отслеживание движущихся объектов и передача данных о их координатах через интерфейс связи, например, посредством серийной связи. Это делает модуль идеальным для использования в различных автоматизированных системах, где требуется управление объектами на основе их положения в пространстве.

Серию умных сервоприводов Dynamixel отличает высокая точность управления и широкий спектр возможностей. Они обладают встроенными датчиками обратной связи, что позволяет им точно контролировать положение и скорость вращения. Благодаря интеграции с модулем TrackingCam3, сервоприводы Dynamixel могут быть управляемы в режиме реального времени на основе данных о расположении объектов, полученных с камеры модуля.

Таким образом, комбинация модуля TrackingCam3 и сервоприводов Dynamixel

предоставляет возможность создания умной системы управления, способной автоматически реагировать на окружающую среду и выполнять задачи с высокой точностью и эффективностью.

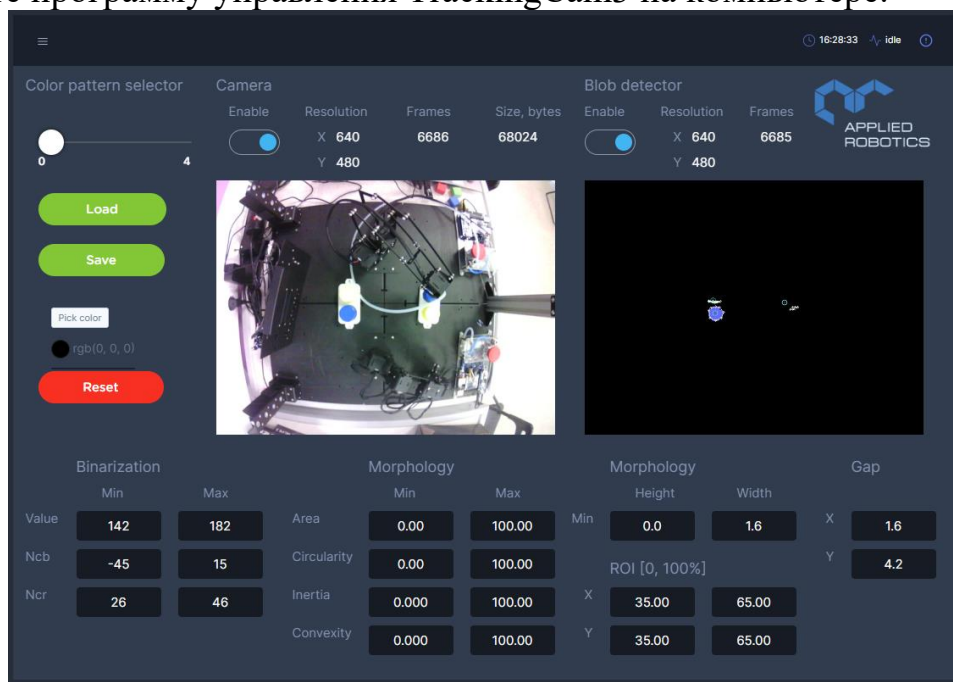
Практическая работа

Задания:

1. Подключите TrackingCam3 к плате OpenCM.

Фото схемы подключения

2. Подключите сервоприводы Dynamixel к контроллеру.
3. Запустите программу управления TrackingCam3 на компьютере.



4. Установите необходимые параметры и настройки подключения.
5. Протестируйте работу модуля и управление манипулятором.

Контрольные вопросы

1. Какие компоненты необходимо подключить для работы с модулем TrackingCam3?
2. Какие виды объектов может распознавать TrackingCam3?

Фрагмент кода № 1

```
#include <DynamixelWorkbench.h>
#define BAUDRATE 1000000
#define CAM_ID 51
#define DEVICE_NAME "3"

struct TrackingCamBlobInfo_t
{
    uint8_t type;
    uint8_t dummy;
    uint16_t cx;
    uint16_t cy;
    uint32_t area;
    uint16_t left;
    uint16_t right;
    uint16_t top;
    uint16_t bottom;
};

DynamixelWorkbench dxl_wb;
unsigned long previousMillis = 0;

void setup()
{
    Serial.begin(57600);
    while (!dxl_wb.init(DEVICE_NAME, BAUDRATE))
    {}
    while (!dxl_wb.ping(CAM_ID))
    {}
}

void loop()
{
    int max_n = 5;
    int n = 0;
    TrackingCamBlobInfo_t blob[10];
    for (int i = 0; i < max_n; i++)
    {
        uint32_t resp[16];
        if (!dxl_wb.readRegister(CAM_ID, 16 + i * 16, 16, resp))
            break;
        int idx = 0;
        blob[i].type = resp[idx++];
        if (blob[i].type == 0xFF)
            break;
        blob[i].dummy = resp[idx++];
        blob[i].cx = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].cy = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].area = (resp[idx] + (resp[idx + 1] << 8)) * 4;
        idx += 2;
        blob[i].left = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].right = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].top = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].bottom = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        n++;
    }

    Serial.println("n = " + String(n));
    for (int i = 0; i < n; i++)
    {
```

```

    Serial.print( String(blob[i].type) + " " + String(blob[i].cx) + " " +
String(blob[i].cy) + " " +
    String(blob[i].area) + " " + String(blob[i].left) + " " +
String(blob[i].right) + " " +
    String(blob[i].top) + " " + String(blob[i].bottom) + "\n");
}

while (millis() - previousMillis < 33)
{
previousMillis = millis();
}

```

Приложение

Фрагмент кода № 2

```

#include <DynamixelWorkbench.h>
#include <Dynamixel2Arduino.h>
#define DXL_SERIAL Serial3
#define DEBUG_SERIAL Serial
#define BAUDRATE 1000000
#define CAM_ID 51
#define DEVICE_NAME "3"
#define DEGREE_COEFF 16383/360
#define ADDR 112
#define LEN 4
const uint8_t DXL_DIR_PIN = 22;
const float DXL_PROTOCOL_VERSION = 2.0;
uint32_t vel = 75
Dynamixel2Arduino dxl (DXL_SERIAL, DXL_DIR_PIN);

struct TrackingCamBlobInfo_t
{
    uint8_t type;
    uint8_t dummy;
    uint16_t cx;
    uint16_t cy;
    uint32_t area;
    uint16_t left;
    uint16_t right;
    uint16_t top;
    uint16_t bottom;
};

DynamixelWorkbench dxl_wb;
int cx, type;
int cy;

void setup() {
    Serial.begin(115200);
    dxl_wb.init(DEVICE_NAME, BAUDRATE);
    dxl_wb.ping(CAM_ID);
    dxl.begin(1000000);
    dxl.setPortProtocolVersion(DXL_PROTOCOL_VERSION);
    dxl.torqueOff(1);
    dxl.torqueOff(2);
    dxl.setOperatingMode(1, OP_POSITION);
    dxl.setOperatingMode(2, OP_POSITION);
    dxl.torqueOn(1);
    dxl.torqueOn(2);
    dxl.write(1, ADDR, (uint8_t*)&vel, LEN, 10);
    dxl.write(2, ADDR, (uint8_t*)&vel, LEN, 10);
}

void loop()
{
    read_cam();
    if (type == 1)
    {
        Serial.println(type);
    }
}

```

```

    dxl.setGoalPosition(1, 11000);
    dxl.setGoalPosition(2, 8000);
    delay(2000);
}
if (type == 2)
{
    Serial.println(type);
    dxl.setGoalPosition(1, 6000);
    dxl.setGoalPosition(2, 8000);
    delay(2000);
}
else
{
    Serial.println(type);
    dxl.setGoalPosition(1, 8500);
    dxl.setGoalPosition(2, 8000);
    delay(2000);
}
}

void read_cam()
{
    int ax_n = 1;
    int n = 0;
    TrackingCamBlobInfo_t blob[10];
    for (int i = 0; i < ax_n; i++)
    {
        uint32_t resp[16];
        if (!dxl_wb.readRegister(CAM_ID, 16 + i * 16, 16, resp))
            break;
        int idx = 0;
        blob[i].type = resp[idx++];
        if (blob[i].type == 0xFF)
            break;
        blob[i].dummy = resp[idx++];
        blob[i].cx = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        cx = blob[i].cx;
        blob[i].cy = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        cy = blob[i].cy;
        blob[i].area = (resp[idx] + (resp[idx + 1] << 8)) * 4;
        idx += 2;
        blob[i].left = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].right = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].top = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        blob[i].bottom = resp[idx] + (resp[idx + 1] << 8);
        idx += 2;
        n++;
        type = blob[i].type;
    }
    Serial.println("n = " + String(n));
    for (int i = 0; i < n; i++)
    {
        Serial.print( String(blob[i].type) + " " + String(blob[i].cx) + " " +
String(blob[i].cy) + "\n");
    }
    delay(500);
}
}

```