Решение задачи управления роботом на трехколесной всенаправленной платформе

Арсений Ярмолинский

Май 2024

Содержание

1 Введение

Одна из важнейших задач управления роботами в лиге Robocup SSL является точное позиционирование робота на поле для захвата мяча, точных ударов и приема пасов. Большая часть команд, включая нашу команду SPbUnited в настоящее время использует прямое управление скоростью каждого робота с центрального сервера [?]. Это добавляет ненужные задержки и ошибки в позиционировании.

Наша команда предлагает альтернативный способ управления роботами через задание конечных координат для каждого робота. Для реализации такого подхода требуется решить ряд задач, в том числе задача локальной одометрии по энкодерам и инерциальным датчикам для определения положения робота в глобальной системе координат с высокой точностью и быстродействием.

2 Постановка задачи

В настоящей работе рассматривается задача одометрии для трехколесного всенаправленного робота команды Eagles (рис. 1b).

Кинематическая схема робота приведена на рис. 2

Известны геометрические параметры робота и углы поворотов каждого из колес.

3 Система управления

Система управления роботом разбита на несколько уровней. Общая архитектура программы роботов представляет собой архитектуру SPA (Sense-Plan-Act) (рис. 3).

Главный цикл программы разбит на три блока:

1. *Sense*: обновление всех устройств ввода и построение математической модели окружения;

(a) Команда SPbUnited (b) Команда Eagles

Рис. 1: Внешний вид робота для Robocup SSL

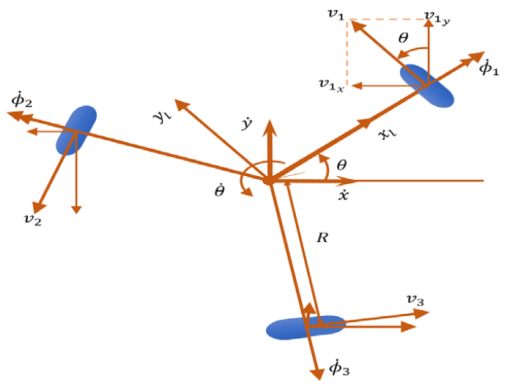

Рис. 2: Кинематическая схема трехколесного робота

Рис. 3: Иллюстрация архитектуры SPA

2. *Plan*: создание плана дейсвтий на основе текущих данных и расчёт всех управляющих воздействий и логики работы робота;

3. *Act*: выполнение плана, а именно выдача управляющих воздействий на исполнительные узлы.

Для реализации обновления устройств ввода и вывода, драйверы данных устройств записаны в соответствующие массивы (Листинг 1).

Обновление устройст соответствующего типа выполняется вызовом функции `updateIN` для устройств ввода и `updateOUT` для устройств вывода.

```
// Input drivers
Updatable *input[] =
{
    &buttonChannelPlus,
    &buttonChannelMinus,
    &buttonEnter,
    &batteryVoltage,
    &ballSensor,
    &nrf,
    &imu
};

// Output drivers
Updatable *output[] =
{
    &motor1,
    &motor2,
    &motor3,
    &kicker,
    &indicator
};

// Peripheral
#define INPUT_N (sizeof(input) / sizeof(input[0]))
#define OUTPUT_N (sizeof(output) / sizeof(output[0]))

// Update all input perripheral
ERROR_TYPE updateIN()
{
    ERROR_TYPE error = NO_ERRORS;
```

```
31
32      // Update everything except indicator
33      for (int i = 0; i < INPUT_N; i++)
34      {
35          error |= input[i]->update();
36      }
37
38      return error;
39  }
40
41  // Update all output perripheral
42  ERROR_TYPE updateOUT()
43  {
44      ERROR_TYPE error = NO_ERRORS;
45
46      // Update everything except indicator
47      for (int i = 0; i < OUTPUT_N; i++)
48      {
49          error |= output[i]->update();
50      }
51
52      return error;
53  }
```

Листинг 1: Обработка драйверов устройств ввода и вывода

Обработчик каждого устройства представляет собой класс, наследуемый от интерфейса `Updatable`, который определяет обязательный для реализации метод `Update()`, используемый для обновления соответствующего устройства ввода-вывода (Листинг 2).

```
1  class Updatable
2  {
3  public:
4      ERROR_TYPE virtual update() = 0;
5  };
```

Листинг 2: Интерфейс `Updatable`

3.1 Управление движением

Управление движением робота разделяется на несколько уровней.

Самый нижний уровень - управление двигателями постоянного тока. На роботе установлено три ДПТ с квадратурными энкодерами (рис. 4), и каждый двигатель управляется независимо.

Для управления двигателями был реализован класс `Motor`, который определяет методы для управления двигателем, такие как задание требуемой скорости, определение его текущей скорости и угла поворота. Управление двигателем осуществляется по скорости.

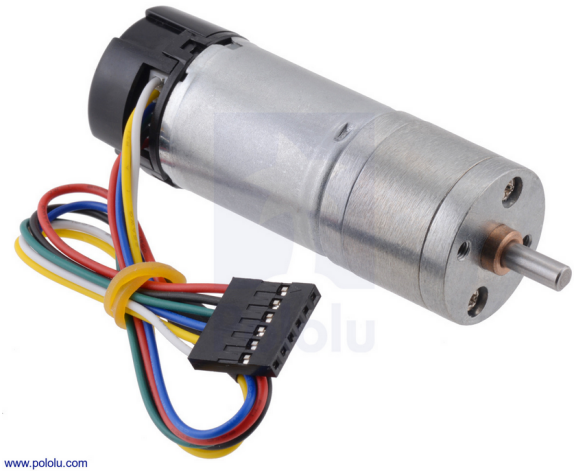

Рис. 4: Мотор 20.4:1 Metal Gearmotor 25Dx65L mm HP 12V with 48 CPR Encoder

3.1.1 Выдача напряжения

Самый нижний уровень - это функция выдачи напряжения на двигатель. Для выдачи соответствующего напряжения информация о текущем напряжении питания аккумулятора комбинируется с требуемым напряжением на двигателе и преобразуется в значения ШИМ, которые выдаются на цифровые пины микроконтроллера.

Для защиты редуктора от резких ударных перекладок реализована возможность добавления ограничения на скорость изменения напряжения, подаваемого на двигатель (Листинг 3).

Использование такого интерфейса выдачи управляющего воздействия в отличие от управления напрямую значениями ШИМ позволяет абстрагироваться от технической реализации ШИМ на микроконтроллере и драйвера мотора, что позволяет не менять настройку системы управления при установке другого двигателя/напряжения питания/микроконтроллера/драйвера.

```
void Motor::applyU(float u)
{
    u *= motorDir;
    if(u == constrain(u, -moveU, moveU))
    {
        u = 0;
    }

    float slowed_u = UchangeLimiter.tick(u);
    int pwm = slowed_u / maxU * 255;

    if (pwm > 255)
        pwm = 255;
    if (pwm < -255)
        pwm = -255;

    if (pwm >= 0)
    {
        analogWrite(IN2, 255);
        analogWrite(IN1, 255 - pwm);
    }
    else
    {
        analogWrite(IN2, 255 + pwm);
        analogWrite(IN1, 255);
    }
}
```

Листинг 3: Функция выдачи напряжения на двигатель

3.1.2 Квадратурный инкрементальный энкодер

Используемые двигатели постоянного тока оборудованы на задней части вала квадратурным инкрементальным энкодером, с помощью которого есть возможность замкнуть обратную связь по скорости и

положению мотора (рис. 5).

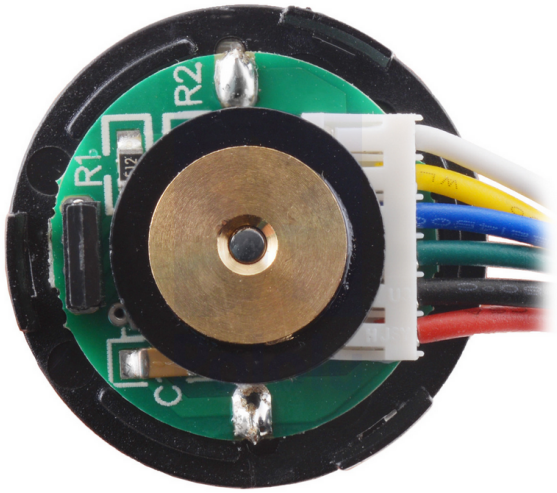

Рис. 5: Внешний вид энкодера на моторе

Инкрементальный энкодер представляет собой устройство, генерирующее импульсы в соответствии с текущим углом поворота вала двигателя. За счет чтения этих импульсов можно определить направление вращения двигателя, а также его текущий угол поворота (рис. 6).

Для подсчета количества импульсов используются pin change interrupts на микроконтроллере. Провода A и B соответствующего энкодера подключены к соседним пинам внутри одного порта, что позволяет быстро считывать их состояние в обработчике прерывания.

Определение направления вращения двигателя было реализовано следующим образом. Сигналы A и B можно скомбинировать в двухбитное целое число. В зависимости от текущей фазы поворота двигателя - это число может принимать значения 00, 01, 11 и 10. Из предыдущего и текущего значения фазы можно однозначно определить в какую сторону был повернут вал двигателя.

Для решения этой задачи была составлена таблица `ett` (encoder transition table, табл. 1), в которой для каждой комбинации преды-

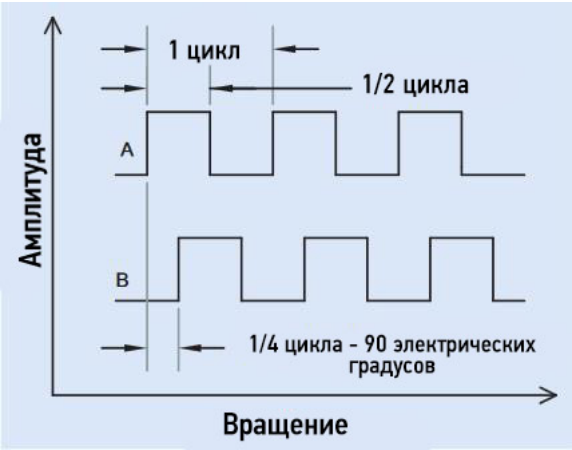

Рис. 6: Вид сигнала с инкрементального энкодера

дущего значения и текущего значения фазы энкодера определяется направление инкремента соответствующее данной комбинации. Таким образом, в обработчике прерываний для определения направления проворота вала двигателя достаточно обратится в таблицу и прочитать соответствующее значение (Листинг 4).

Таблица 1: `encoder transition table`

	00	01	11	10
00	0	-1	0	1
01	1	0	-1	0
11	0	1	0	-1
10	-1	0	1	0

```
1 void Motor::interruptHandler()
2 {
3     uint8_t enc = ((*ENC_PORT) & ENC_MASK) >> ENC_SHIFT;
4     counter += ett[prevEnc][enc];
5     prevEnc = enc;
6 }
```

Листинг 4: Обработчик прерываний энкодера

3.1.3 Вычисление скорости вала

Для замыкания обратной связи по скорости двигателя был реализован вычислитель скорости привода. Вычислитель сконструирован методом переоборудования передаточной функции следующего вида:

$$W_f(s) = \frac{s}{Ts + 1}, \tag{1}$$

представляющей из себя инерционно-дифференцирующее звено. Постоянная времени принята равной $T = 2T_s$, где $T_s = 6$мс - период квантования системы.

3.1.4 Регулирование скорости

Общая структура контура скорости двигателя представлена на рис. 7.

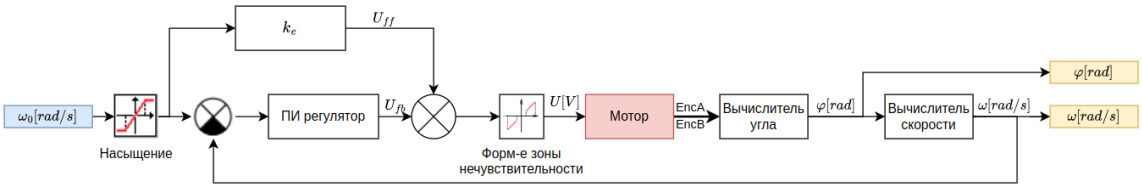

Рис. 7: Система управления скоростью мотора

Для замыкания обратной связи по скорости вычисленная текущая угловая скорость двигателя подается на вход ПИ-регулятора скорости, коэффициенты которого рассчитаны в соответствии с настройкой на модульный оптимум. Помимо ПИ-регулятора, реализовано комбинированное управление с добавкой к напряжению по прямому каналу, рассчитанное с помощью коэффициента усиления привода.

Для устранения писка двигателя при заторможенном роторе на низких скоростях был реализован алгоритм формирования зоны нечувствительности (рис. 8), который обнуляет управляющее воздействие,

если оно входит в порог этой зоны. Граница зоны нечувствительности определена экспериментально как напряжение трогания двигателя постоянного тока, которое для данных двигателей составляет 2 В.

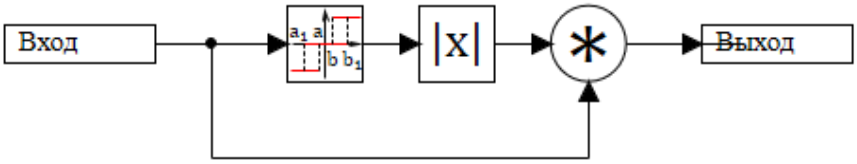

Рис. 8: Алгоритм формирования зоны нечуствительности

3.2 Прямое управление скоростями

Управление роботом происходит через команды, посылаемые по радиоканалу. Управление реализовано через команды скорости в глобальной системе координат.

Команда с сервера представляет из себя вектор из трех значений: $\{v_x, v_y, \omega\}$, где v_x и v_y - требуемые линейные скорости в глобальной системе координат, ω - требуемая угловая скорость робота.

На рис. 2 изображена кинематическая схема робота.

Исходя из геометрии робота скорости в глобальной СК можно пересчитать в требуемые скорости колес можно пересчитать используя следующие соотношения:

$$\begin{bmatrix} \dot{\phi}_1 \\ \dot{\phi}_2 \\ \dot{\phi}_3 \end{bmatrix} = \frac{1}{r} \begin{bmatrix} -\sin(\theta) & \cos(\theta) & R \\ -\sin(\theta+\alpha_2) & \cos(\theta+\alpha_2) & R \\ -\sin(\theta+\alpha_3) & \cos(\theta+\alpha_3) & R \end{bmatrix} \begin{bmatrix} \cos(\theta) & 0 & 0 \\ 0 & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix} \quad (2)$$

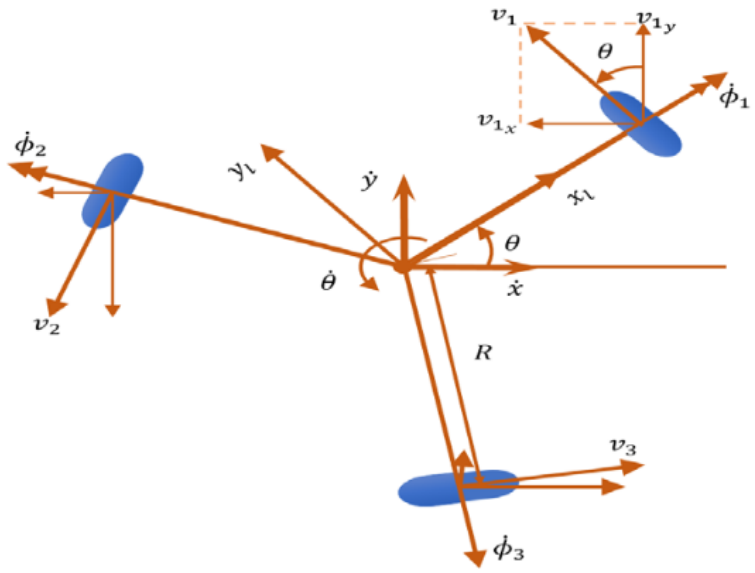

Рис. 9: Кинематическая схема робота

3.2.1 Стабилизация по гироскопу

Для обеспечения курсовой устойчивостью робота была реализована обратная связь по установленному на его борту гироскопу (рис. 10). Дело в том что при резких разгонах и торможениях разность моментов на разных двигателях может привести к нежелательному развороту робота от целевого положения. Удержание угла с использованием гироскопа позволит избежать подобных проблем.

Для замыкания обратной связи была реализована пропорциональная обратная связь по интегральной оценке разности целевого угла и текущего. Структурная схема системы управления представлена на рис. 11.

Рис. 10: Модуль гироскопа-акселерометра MPU9250

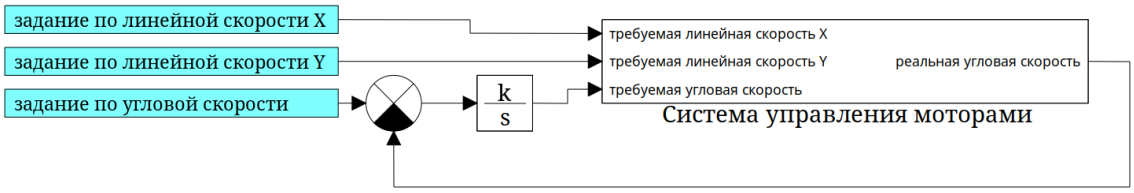

Рис. 11: Схема системы управления с обратной связью по гироскопу

4 Результаты

В результате проделанной работы была получена система управления трёхколёсным роботом на всенаправленные платформе для применения в соревнованиях Robocup SSL. Система управления позволяет управлять линейными и угловой скоростью робота с поддержанием курсовой устойчивости для повышения качества алгоритмов управления.

5 Будущие улучшения

В числе будущих задач числится разработка алгоритмов локальной одометрии робота и управление роботом по положению в глобальной системе координат.