Размещено на http://www.allbest.ru/

Разработка ПИД–регулятора для движения по линии робота Lego Ev3 с использованием программной библиотеки RubiRobotLib **на Posix–совместимой ОС**

Размещено на http://www.allbest.ru/

Введение

Данная курсовая работа посвящена разработке ПИД-регулятора для движения по линии робота Lego EV3 с использованием программной библиотеки RubiRobotLib на Posix-совместимой ОС.

Актуальность темы курсовой работы связана с интенсивным развитием робототехники как в стране, так и во всём мире. Ежегодно проводятся различные соревнования по робототехнике. Самый популярный вид соревнований по робототехнике – это езда робота по линии. ПИД – регулятор обеспечивает наилучшую результативность прохождения трассы. А библиотека RubiRobotLib в связке с проектом ev3dev, предоставляет высокий функционал и удобство в программировании роботов Lego EV3. Таким образом данная разработка покажет уровень возможностей робота в данном виде соревнований.

Целью курсовой работы является разработка ПИД – регулятора для движения линии робота Lego EV3 с использованием программной библиотеки RubiRobotLib на Posix-совместимой ОС.

Задачи:

1) Ознакомиться с необходимой документацией по проекту ev3dev и программной библиотеке RubiRobotLib.
2) Изучить принципы пропорционального, интегрального и дифференциального регулирования.
3) Разработать поэтапно ПИД – регулятор с использованием библиотеки RubiRobotLib написанной под язык программирования высокого уровня Free Pascal для езды по линии робота Lego EV3.
4) Разработать линейный алгоритм с процедурной организацией для прохождения полигона «Мастер» в основе которого лежит ПИД – регулятор и протестировать данный алгоритм, вшитый в робота на полигоне «Мастер».

Размещено на http://www.allbest.ru/

Размещено на http://www.allbest.ru/

1.Теоретический основы ПИД–регулирования

1.1 Основы ПИД регулирования

Основная задача – разработать ПИД регулятор. Для этого нужно обратиться к математике. Т. к. основы ПИД регулирования описаны именно её формулами.

ПИД регулятор – это совокупность пропорционального, интегрального и дифференциального регуляторов. Каждый из них, отвечает за определённые опции. Основным считается пропорциональное звено т.к. для любой системы важно быстродействие, что и обеспечивает П – регулятор.

Для того, чтобы разработать ПИД – регулятор и внедрить его в робота, необходимо знать принципы ПИД – регулирования и как совокупность регуляторов, и как каждое отдельное звено.

ПИД -регулятор:

1) **Пропорциональный регулятор**

$$P(t) = Kp * e(t) \quad (1)$$

где

K_p - коэффициент пропорциональный;

e(t) – значение ошибки рассогласования.

Пропорциональный регулятор не имеет «памяти», следовательно, значение выходного сигнала не зависит от предыдущего состояния всей системы. ошибка рассогласования на текущей итерации, умноженная на коэффициент пропорциональный, передается на выход. Выходной сигнал компенсирует отклонение регулируемого параметра. Сигнал тем больше, чем больше ошибка рассогласования. При ошибке равной 0, сигнал на выходе

Размещено на http://www.allbest.ru/

тоже равен 0.

Пропорциональная составляющая не способна компенсировать ошибку полностью. Выходной сигнал в $\mathbf{K_p}$ раз больше ошибки. Если ошибка рассогласования равна 0, то и выходной сигнал регулятора равен 0. А тогда и компенсировать нечем.

Поэтому в пропорциональных регуляторах всегда существует так называемая статическая ошибка. Это бесчисленные незначительные колебания уменьшить которые можно за счет увеличения коэффициента **Kp**, но это может привести к снижению устойчивости системы и даже к автоколебаниям. П – регулятор не способен избавиться от статической ошибки, он лишь способен сделать её незначительной.

Недостатки:

a) наличие статической ошибки регулирования;

b) невысокая устойчивость при увеличении коэффициента.

Преимущества:

a) Высокая скорость регулирования. Реакция пропорционального регулятора на ошибку рассогласования ограничена только временем дискретизации системы.

Регуляторы, работающие только по пропорциональному закону, применяют редко так как статическая ошибка сильно влияет на устойчивость всей системы, да и П – регулятор статичен, он не подстраивается под значение предыдущих ошибок на предыдущих итерациях.

Главная задача пропорциональной составляющей в ПИД регуляторе – повысить быстродействие.

2) **Интегрирующая составляющая**

$$I(t) = Ki \int e(t)dt \quad (2)$$

Размещено на http://www.allbest.ru/

Пропорциональна интегралу ошибки рассогласования. С учетом временной дискретности регулятора можно написать так:

Размещено на http://www.allbest.ru/

$$I(t) = I(t-1) + Ki * e(t) \quad (3)$$

где

K_i – коэффициент интегрирования;

e(t) – значение ошибки рассогласования;

$I(t_{-1})$ – это значение интегрирующего звена на предыдущей итерации.

Ошибка рассогласования умножается на коэффициент и прибавляется к предыдущему значению интегрирующего звена. Т. е. выходной сигнал все время накапливается и со временем увеличивает свое воздействие на объект. Таким образом, ошибка рассогласования полностью компенсируется даже при малых значениях ошибки и коэффициента $\mathbf{K_i}$. В установившемся состоянии выходной сигнал регулятора полностью обеспечивается интегрирующей составляющей.

Недостатки:

a) низкое быстродействие;

b) посредственная устойчивость.

Достоинства:

a) Способность полностью компенсировать ошибку рассогласования (статическую) при любом коэффициенте усиления.

На практике часто используют интегрирующие регуляторы (только интегрирующая составляющая) и пропорционально-интегрирующие (интегрирующая и пропорциональная составляющие).

Главная задача интегрирующего звена в ПИД регуляторе – компенсация статической ошибки, обеспечение высокой точности регулирования.

3) **Дифференцирующая составляющая**

Размещено на http://www.allbest.ru/

$$D(t)=Kd*de(t)/dt \quad (4)$$

$$D(t)=Kd*(e(t)-e(t-1)) \quad (5)$$

где

K_d – коэффициент дифференцирования;

e(t) – значение ошибки рассогласования;

$e(t_{-1})$ – значение ошибки рассогласования на предыдущей итерации.

Дифференцирующая составляющая пропорциональна скорости изменения ошибки рассогласования. Своеобразный показатель ускорения ошибки рассогласования. Дифференцирующая составляющая предсказывает отклонения регулируемого параметра в будущем и противодействует этому отклонению. Как правило, она компенсирует запаздывания, воздействия регулятора на объект и повышает устойчивость системы.

Она показывает, насколько изменилось значение ошибки рассогласования за одну временную единицу дискретности регулятора.

Регуляторов, состоящих из единственного дифференцирующего звена, не бывает, т. к. дифференцирующее звено без других не на что не способно.

Главная задача дифференцирующего звена в ПИД регуляторе – повышение устойчивости.

1.2 Используемый инструментарий

Для разработки приложения необходима платформа, в качестве платформы для разрабатываемого приложения была выбрана Linux подобная система, поддерживающая POSIX стандарт который, является платформенно-независимым системным интерфейсом для компьютерного окружения.

Размещено на http://www.allbest.ru/

1) **POSIX** – это стандарт IEEE, описывающий системные интерфейсы для открытых операционных систем, в том числе оболочки, утилиты и инструментарии. Так же, согласно POSIX, стандартизированными являются задачи обеспечения безопасности, задачи реального времени, процессы администрирования, сетевые функции и обработка транзакций. Стандарт базируется на UNIX-системах.

Задачи Posix:

1) Определение и унификация интерфейсов в процессе проектирования, а не в момент их реализации.

2) Облегчение переноса кода прикладных программ на другие платформы.

3) Определение необходимого минимума интерфейсов прикладных программ для увеличения скорости создания, одобрения и утверждения документации.

4) Сохранение и учёт всех главных, созданных ранее и используемых прикладных программ.

5) Развитие стандартов в направлении обеспечения коммуникационных сетей, распределения обработки данных и защиты информации.

2) **Программная библиотека RubiRobotLib. Проект ev3dev**

RubiRobot — это программная библиотека, написанная под язык программирования высокого уровня Free Pascal, позволяющая выполнять огромное количество манипуляций с роботом на платформе Ev3, для создания различных алгоритмов поведения.

Данная библиотека целиком и полностью зависит от проекта ev3dev. Ev3dev – это проект, который позволяет накатить прошивку в виде Linux Debian на блок Lego Ev3. Именно данная прошивка позволяет программировать робота на более прикладном уровне, используя языки

Размещено на http://www.allbest.ru/

программирования. Установив на блок Ev3 дистрибутив Linux Debian посредством microSD карты и утилиты Etcher, и поставив соответствующее ПО RubiRobotManager на ПК под управлением ОС Linux, можно взаимодействовать с роботом с помощью ssh-сессий, программируя и компилируя все программы прямиком на самом блоке Ev3, что так же обеспечивает комфорт в разработке различных алгоритмов поведения робота.

Документация по библиотеке находиться на официальном сайте автора: http://rubirobot.ru/. В данной документации описаны все возможности по инициализации модулей робота (сервоприводов, различных датчиков), по функциям и возможностям контроля модулями и небольшая инструкция по установке всего необходимого для начала эксплуатации.

3) Free Pascal

Free Pascal, также известный как FPC (free pascal compiler) - свободно распространяемый в исходных текстах 32-разрядный кроссплатформенный компилятор языка **Pascal.**

Pascal - это чисто процедурный язык программирования, часто использующийся для обучения структурному программированию.

Особенностями языка являются строгая типизация и наличие средств структурного (процедурного) программирования. Pascal был одним из первых таких языков.

4) Lego Ev3

Lego Ev3 – вычислительный блок, у которого имеется arm процессор, 8 разъёмов для подключение модулей (сервоприводов и различных датчиков), не большой чёрно-белый дисплей с клавишами управления, usb выход для подключения wifi адаптера и разъём для microSD карточки.

Данный блок распространён во всём мире, именно на нём подавляющее большинство участвуют на соревнованиях по робототехнике. Не имеет явных конкурентов.

Размещено на http://www.allbest.ru/

1.3 Принцип настройки ПИД – регулятора

ПИД – регулятор в разных случаях может работать по-разному, хоть он и очень эффективен, вся эффективность может улетучиться в миг. Поэтому, качество регулирования ПИД регуляторов в значительной мере зависит от того, насколько оптимально выбраны коэффициенты. Коэффициенты ПИД регулятора определяются на практике путем экспериментального подбора.

Составляющие ПИД регулятора настраиваются отдельно:

- Отключается интегрирующее(ki) и дифференцирующее(kd) звенья и выбирается коэффициент пропорционального(kp) звена. Если регулятор пропорционально-интегрирующий (отсутствует дифференцирующее звено), то добиваются полного отсутствия колебаний на переходной характеристике. При настройке регулятора на высокое быстродействие колебания могут остаться. Их можно попытаться скомпенсировать дифференцирующим звеном.

a) Далее подключается дифференцирующее звено. Его коэффициентом стремятся убрать колебания параметра регулирования. Если не удается, то уменьшают пропорциональный коэффициент.

b) За счет интегрирующего звена убирают остаточную ошибку рассогласования.

Настройка ПИД регулятора носит итерационный характер. Т.е. пункты подбора коэффициентов могут многократно повторяться до тех пор, пока не будет достигнут приемлемый результат.

Благодаря высоким характеристикам и универсальности ПИД регуляторы широко применяются в системах автоматизации производства.

Общая формула ПИД – регулятора:

Размещено на http://www.allbest.ru/

$$\mathbf{PID = K_p * e(t) + I_{(t-1)} + K_i * e(t) + K_d * (e(t) - e(t-1))} \quad \mathbf{(6)}$$

Таким образом, информация об основах ПИД – регулирования и необходимом оборудовании, документация о проекте ev3dev и RubiRobotLib плавно подводит к началу разработки самого ПИД – регулятора.

Размещено на http://www.allbest.ru/

2. Разработка и экспериментальное использование ПИД – регулятора

2.1 Подготовка к разработке ПИД – регулятора

Для подготовки к разработке ПИД – регулятора необходимо установить и настроить прошивку от проекта ev3dev на вычислительный блок Lego Ev3. (вся необходимая информация об установке прошивки находиться на сайте автора библиотеки rubirobot.ru). После установки прошивки подключить блок Lego Ev3 по USB порту, или по беспроводному соединению, если есть такая возможность. Открыть среду разработки с поддержкой FPC.

Для начала нужно инициализировать модули библиотеки RubiRobotLib для того, чтобы появилась возможность производить манипуляции с датчиками робота. (рис. 1).

Рис. 1. Инициализация модулей.

Инициализация всех необходимых переменных, которые будут задействованы в реализации ПИД – регулятора. Можно сразу продумать структуру алгоритма ПИД – регулирования, можно по ходу, добавлять новые переменные. (Рис. 2).

Размещено на http://www.allbest.ru/

Рис. 2. Инициализация переменных. (в т. ч. переменных типа TEv3).

Размещено на http://www.allbest.ru/

Стоит на данном этапе обратить внимание на нижние переменные, у которых указан тип TEv3ColorSesor и TEv3LargeMotor. Это первые механизмы библиотеки RubiRobotLib, которые необходимо использовать для дальнейшей работы с сервоприводами и датчиками.

Для того, что бы функции программной библиотеки RubiRobotLib были инициализированы и заработали необходимо в начале программы необходимо прописать **ev3init();** Это своего рода инициализация самой библиотеки и её функций в FPC.

Для того, чтобы робот ездил по линии, необходимы следующие модули:

a) Два мотора (больших или средних).

b) Два датчика цвета, которые способны работать в двух режимах, считывать цвет, считывать яркость (от 0 до 100). Для реализации алгоритма езды по чёрной линии, более эффективно будет использовать второй режим (датчик освещённости).

Для того, чтобы робот ездил, необходимо, по мимо имеющихся и подключенных в соответствующие разъёмы модулей, эти самые модули инициализировать. (Рис. 3).

Рис. 3. Инициализация библиотеки и датчиков(портов).

Так же можно написать простую функцию движения робота под названием move. Перед телом программы нужно её инициализировать. (Рис. 4). Можно обойтись и без неё. В этой функции описан метод передачи моторам, мощности.

Размещено на http://www.allbest.ru/

Рис. 4. Инициализация функции move.

Движение моторов осуществляется с помощью вспомогательного параметра Run(мощность), который пишется через точку.

Последнее действие в подготовке к написанию программы, это считывание показаний с датчиков освещённости. Этого можно достичь с помощью параметра reflect. (Рис. 5).

Рис. 5. Считывание показаний с датчиков.

Выводить значения датчиков на экран не обязательно, но это очень полезно в плане отладки программы.

2.2 Этапы разработки ПИД–регулятора

Для разработки ПИД – регулятора нужна выстроить определённый поэтапный план. ПИД – регулятор — это совокупность трёх регуляторов. Соответственно, для разработки и реализации ПИД – регулятора, по мимо, понимания принципов ПИД регулирования, как совокупность трёх звеньев,

Размещено на http://www.allbest.ru/

так и по отдельности, необходимо так же разрабатывать и регулятор, поэлементно.

Этапность такова:

a) Пропорциональный регулятор;

b) Пропорционально-интегральный регулятор;

c) Пропорционально-интегрально-дифференциальный регулятор.

2.3 Разработка ПИД–регулятора

1) Реализация пропорционального регулятора

Для реализации пропорционально регулятора необходимо инициализировать переменные (Рис. 6), которые будут отвечать за ошибку рассогласования, коэффициент пропорционального звена и мощность, с которой будут работать двигатели. (err, kp, v).

Рис. 6. Инициализация переменных П – регулятора

Далее необходимо инициализировать значение с самих датчиков. Иными словами, получить эти значения. В необходимой для езды по чёрной линии и прохода полигона «Мастер» конструкции робота, имеются два датчика освещённости, которые были присвоены двум переменным: colresult1 и colresult2 (Левый и правый соответственно). И два больших мотора, с которых должно считывать мощность. (LMSpeed, RMSpeed). По необходимости можно выводить значения датчиков и моторов в дисплей робота, либо компьютера. (Рис. 7).

Размещено на http://www.allbest.ru/

Рис 7. Инициализация значений, считываемых с датчиков.

Размещено на http://www.allbest.ru/

Далее необходимо реализовать сам П – регулятор. Начать стоит с ошибки рассогласования. В случае с роботом, который будет ездить по чёрной линии, ошибкой рассогласования будет являться разница датчиков освещённости. Если значение с этих датчиков равны друг-другу, то и ошибка будет равна нулю. Если значения с датчиков будут разница, то и ошибка будет иметь определённое значение, которое будет увеличивать силу противления этой самой ошибке. Записать это стоит так: err:=(colresult1-colresult2); Вычитаем из значений первого датчика. Значения второго.

Далее необходимо подставить всё вышесказанное в формулу, описанную в теоретической части. Значение K_p инициализировано ещё при объявлении этих самых переменных.

Подставив форму в формулу, необходимо, чтобы результат этой формулы менял мощность самих моторов. Что бы робот понимал, когда ему нужно поворачивать налево, а когда направо. Всё этом можно записать в несколько строчек. (рис. 8).

Рис. 8. Механизм П - регулирования.

После проделывания вышеописанных действий, при выходи из цикла, следует остановить моторы. Цикл необходимо создать бесконечным, если роботу необходимо ездить по замкнутому кругу. Выход из цикла можно запрограммировать на нажатие какой-либо клавиши. Полная реализация П – регулятора. (Рис. 9).

Размещено на http://www.allbest.ru/

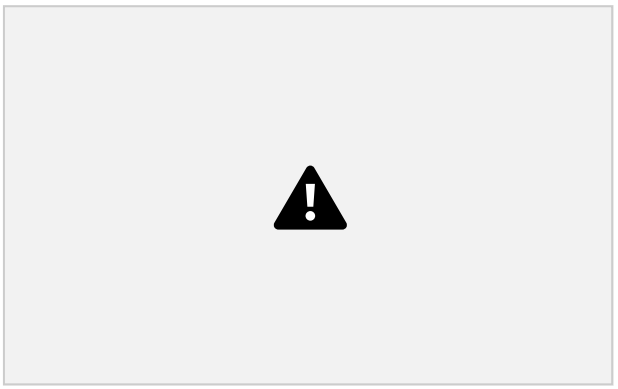

Рис 9. П – регулятор.

2) **Реализация пропорционально-интегрального регулятора**

Для ПИ – регулятора, необходимо инициализировать несколько новых переменных (it, itold, ki). It – значение всей интегрального звена. Itold – значение интегрального звена на предыдущей итерации. Ki – коэффициент интегрирования. (Рис. 10).

Рис. 10. Переменные для ПИ – регулятора.

Ошибка рассогласования вычисляется тем, же способом, что и в П – регуляторе, и в последующих модернизациях регулятора, способ её вычисления останется неизменным.

Далее необходимо полученные значения подставить в формулу и передать значения этой формулы в сами двигателя. (Рис. 11).

Размещено на http://www.allbest.ru/

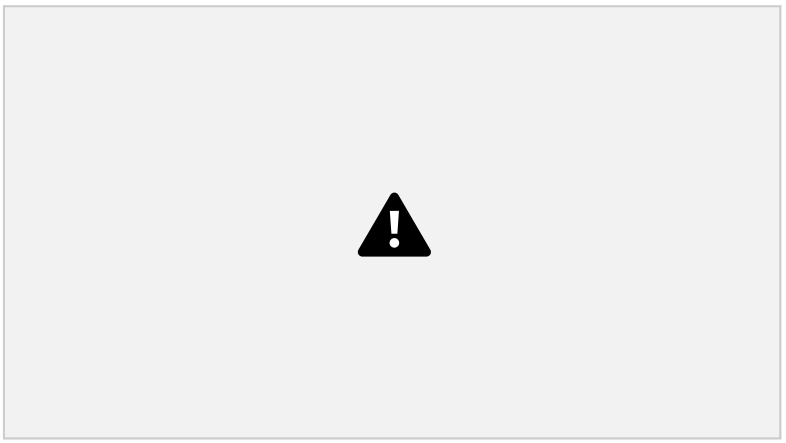

Рис 11. Механизм ПИ – регулятора.

После механизма ПИ – регулятора необходимо остановить моторы. Цикл тот же что и в П – регуляторе, цикл будет так же не изменен до реализации линейного алгоритма для прохождения трассы «Мастер», т. к. там в ход идёт процедурная реализация. Реализация ПИ – регулятора целиком. (Рис. 12).

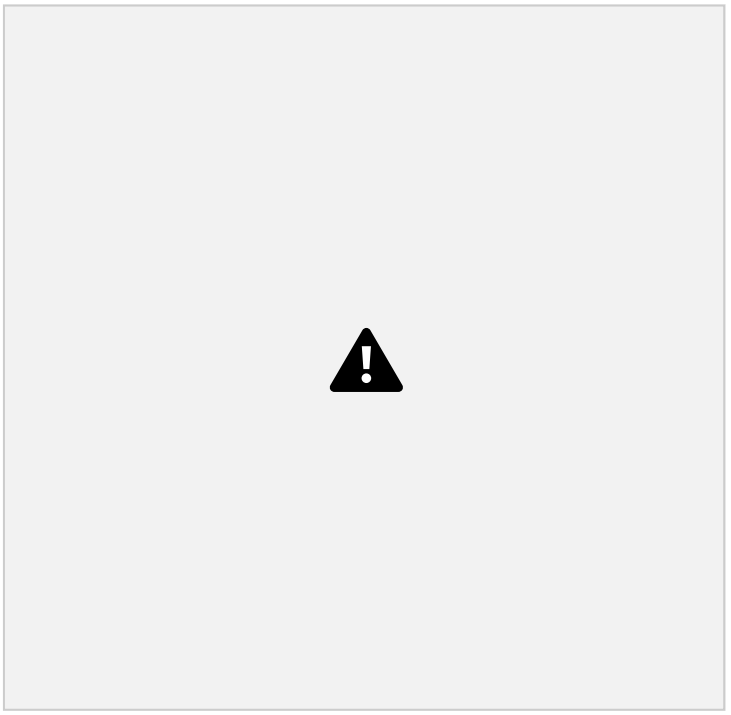

Рис 12. ПИ – регулятор.

Размещено на http://www.allbest.ru/

Осталось реализовать последнее звено, дифференциальное. На этом этапе заканчивается разработка самого ПИД – регулятора.

Для начала вновь следует дополнить список уже существующих переменных новыми (kd, errold).

Рис 13. Переменные ПИД – регулятора.

Kd – коэффициент дифференцирования, errold – ошибка на предыдущей итерации.

После инициализации всех переменных вновь дополняем механизм регулятора. На этот раз добавляем дифференцирующее звено. И передаём значение всего регулятора на мощность моторов. (Рис. 14).

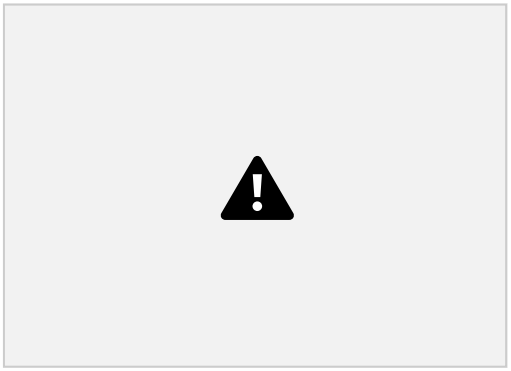

Рис. 14. Механизм ПИД – регулирования.

Для удобства желательно расписать по отдельности каждую формулу, а позже просто сложить полученные результаты трёх формул и передать их моторам. Так же в конце всех действий необходимо не забывать пере

Размещено на http://www.allbest.ru/

присваивать значение переменных it, errold и остановить моторы по выходу из цикла. Реализация ПИД – регулятора целиком (рис. 15.)

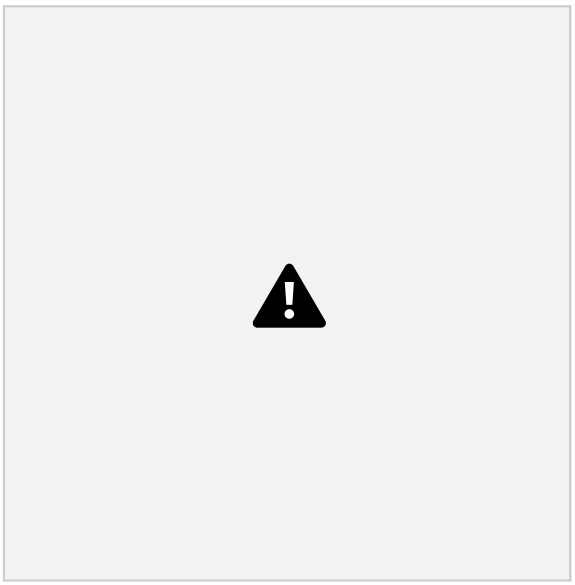

Рис. 15. ПИД – регулятор.

2.4 Разработка линейного алгоритма для прохождения полигонов

После разработки ПИД – регулятора, его можно интегрировать в любой линейный алгоритм, для любой трассы посредством процедур. ПИД – регулятор обеспечит высокую результативность на трассах, при грамотном подборе коэффициентов регулирования.

Для начало следует взять вышеописанный механизм регулирования и заключить в процедуре. Аргументами этой процедуры будут являться, мощность двигателей, три коэффициента регулирования (kp, ki, kd) и время выполнения процедуры. Последний аргумент вариативен, поэтому создать следует две процедуры, одна с ключевым параметром времени, а другая нет. (Рис. 16, 17).

Размещено на http://www.allbest.ru/

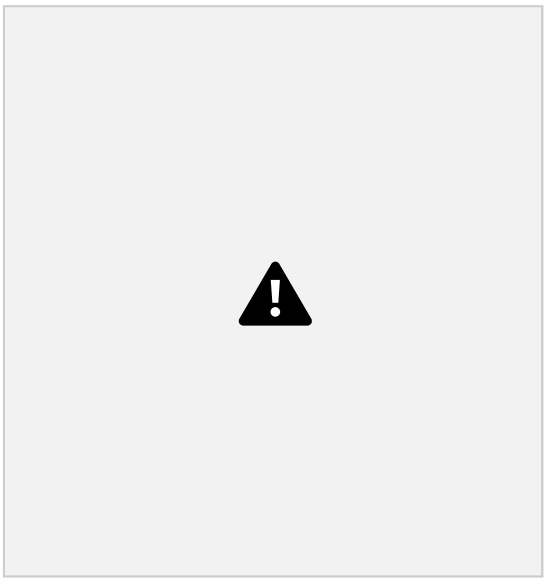

Рис 16. Процедура ПИД – регулирования с параметром времени.

Рис. 17. Процедура ПИД – регулирования без параметра времени.

Если у процедуры с параметром времени, время является условием завершения процедуры, то у процедуры без этого параметра нужно разработать другой механизм её завершение. Механизм, который бы повышал эффективность робота на любой трассе.

Если рассмотреть трассу полигона «Мастер» можно увидеть места, которые при малейших колебаниях, при малейшей ошибке, робот будет проходить не стабильно, особенно на больших скоростях. (рис. 19).

Размещено на http://www.allbest.ru/

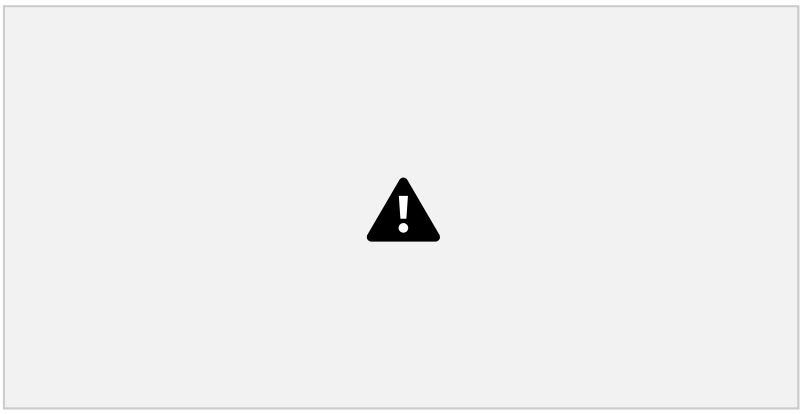

Рис. 19. Полигон «Мастер».

Это перекрёстки. Если робот при малейшем отклонении войдёт на данный участок, то его тряхнёт, что может привести к вылету с трассы. Такие участки эффективно проходить отключая датчики. Для этого подойдёт первая процедура со временем, датчики отключить можно, передав параметрам регулирования значения равные нулю. Регулятор просто не будет реагировать на какие-либо значения датчиков освещённости, что и необходимо на данном участке, не говоря об рекурсивном участке (где цвета линии и фона меняются местами).

У первого датчика лишь один серьёзный недостаток, как не странно он является и его преимуществом, это параметр времени. На некоторых участках трассы это самый эффективный механизм, но при каких-либо сложных участках трассы (перекрёсток, реверсия), необходимо определять, что робот вошёл на данный участок, и в целях безопасности, пройти его отключив датчики. В этом случае выступает вторая процедура, условием выхода которой является нахождения двух датчиков на чёрном цвете, так робот понимает, что он находиться на перекрёстке или на реверсированном участке. При входе на данный участок процедура pidalgo завершается и алгоритм идёт дальше. А дальше будет нужно вновь вызвать процедуру с параметром времени на очень короткий строк, передав параметрам регулирования нули. Так робот пролетит данный участок ровно и без риска вылета за трассу.

Размещено на http://www.allbest.ru/

Можно создать одну процедуру с двумя условиями выхода. Такая процедура будет вариативна, но это на порядок может запутать, и неблагоприятно сказаться на читабельности и понимании кода.

Следует отметить, что пока одна процедура линейного алгоритма выполняется, другие ждут. Это не асинхронный вызов. Следовательно, нужно грамотно расставлять вышеописанный процедуры и учитывать условия выхода из этих самых процедур, иначе они могут привести к вылету с трассы. (Рис. 20).

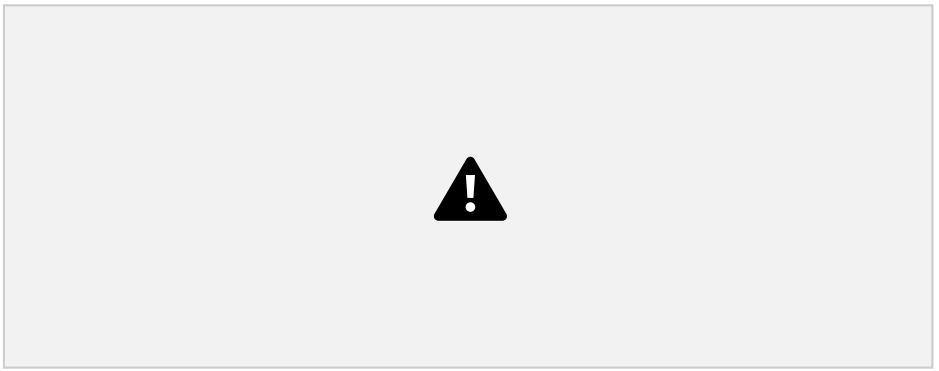

Рис. 20. Не асинхронный вызов процедур линейного алгоритма прохождения полигона.

Следующие процедуры, которые необходимо разработать, это процедуры поворота направо и налево. Программная библиотека RubiRibotLib позволяет совершать повороты по оборотам моторов, по градусам и по времени, но эти все механизмы не выделяются хорошей стабильностью, ведь если робот зайдёт на какой-либо участок не прямо, а чуть наискосок, то робот может пере рулить и вылететь с трассы. Для этого нужно вновь воспользоваться датчиками освещённости. Механизм довольно прост. В зависимости от направления поворота, считывать либо по правому, либо по левому датчику значения и создать линейную процедуру поворота. (Рис. 21, 22).

Размещено на http://www.allbest.ru/

Размещено на http://www.allbest.ru/

Рис. 22. Процедура поворота направо.

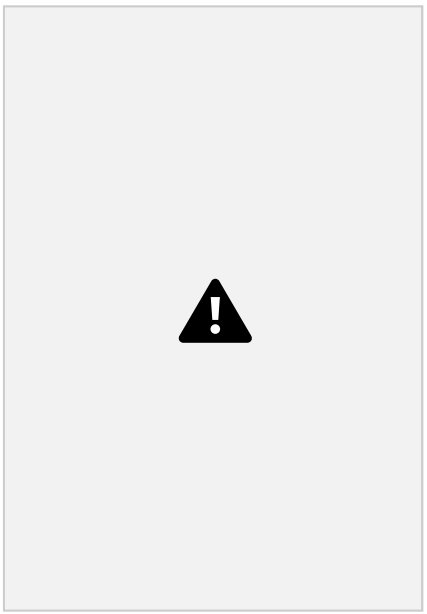

Рис. 21. Процедуры поворота налево.

Вариативность данных процедур колеблется от направления и от того, какой датчик будет ведущим. Но однозначным преимуществом этих процедур является точность поворотов. Робот встаёт ровно (датчиками между чёрной линии).

После разработки вышеперечисленных процедур нужно сконфигурировать сам линейный алгоритм из этих самых процедур. И подобрать приемлемые коэффициенты регулирования для каждого участка трассы. (Рис. 22, 23).

Размещено на http://www.allbest.ru/

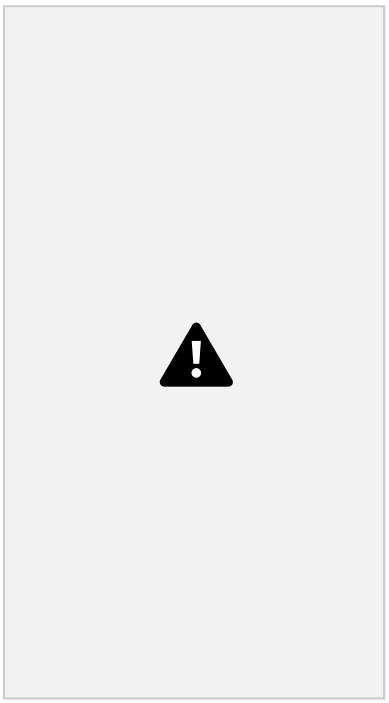

Рис. 22. Линейный алгоритм прохождения полигона «Мастер» с использованием ПИД – регулятора.

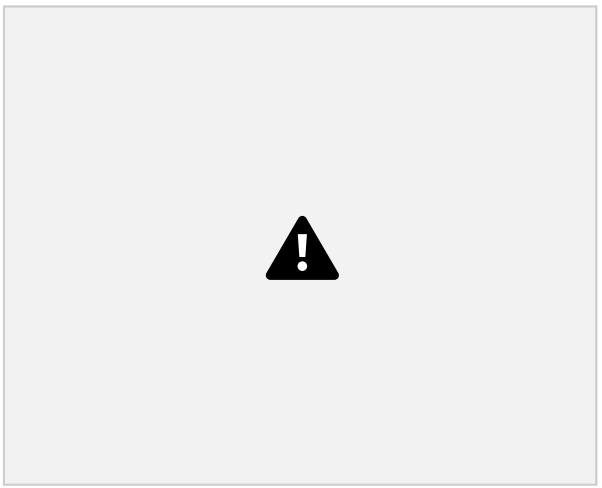

Рис. 23. Линейный алгоритм прохождения полигона «Мастер» с использованием ПИД – регулятора.

Результат данного алгоритма, успешный проход трассы на категорию «Мастер» (Рис. 19) за 30 с небольшим секунд. И это не предел. Доказательства прохода находятся в ленте новостей автора библиотеки RubiRobotLib, rubirobot.ru. (Рис. 24).

Размещено на http://www.allbest.ru/

Размещено на http://www.allbest.ru/

Рис. 24. Доказательство работоспособности программы на сайте rubirobot.ru.

Конечный результат положительный. Разработка ПИД – регулятора завершена успешна, а линейный алгоритм с процедурной организацией ёмко иллюстрирует работоспособность и превосходство ПИД – регуляторов по отношению к другим в робототехнике.

Размещено на http://www.allbest.ru/

Заключение

В рамках данной курсовой работы были разработаны ПИД – регулятор с циклической организацией, линейный алгоритм для успешного прохождения полигона «Мастер» с процедурной конфигурацией и внедрённым в процедурную составляющую ПИД – регулятором на POXIX-совместимой ОС.

При проектировке ПИД – регулятора акцент делался на устойчивость регулятора в эксплуатации. На читабельность и понимание каждой составляющей механизма ПИД – регулирования. При проектировки линейного алгоритма делался акцент на универсальность процедурной реализации механизма под любой полигон и любую трассу с возможностью её дальнейшей модификации. Данный алгоритм, механизм читабелен и очень прост в освоении, что в свою очередь позволит начинающему программисту легко разобраться в принципах ПИД – регулирования и освоить часть необходимых функций программной библиотеки RubiRobotLib, такие как, передача мощности в моторы, считывания показаний с датчиков и моторов, рулевой управление для удобства в поворотах и т. д.

В итоге, в соответствии с поставленной задачей, был достигнут результат, а именно: разработать ПИД – регулятора для движения линии робота Lego EV3 с использованием программной библиотеки RubiRobotLib на Posix-совместимой ОС. После разработки, продемонстрированно на примере полигона и универсального линейного алгоритма с процедурной реализацией прохождения этого полигона с использованием ПИД – регулятора. Результат в 30 с небольшим секунд более чем примелем, если брать в учёт, что задача стояла не в прохождении полигона на скорость, а в демонстрации ПИД – регулятора.

Размещено на http://www.allbest.ru/

Размещено на http://www.allbest.ru/

Список использованных источников

1. Галатенко, В.А. Программирование в стандарте POSIX [Электронный ресурс]: курс лекций: учеб. пособие / В. А. Галатенко; под ред. В.Б. Бетелина. – М.: Интернет-Университет Информационных Технологий, 2004. – 558 с. – Доступ с сайта ЭБС Университетская библиотека ONLINE. – Режим доступа: URL: http://biblioclub.ru/index.php?page=book&id=233064. – 14.12.2018.
2. Основные понятия и идеи стандарта POSIX [Электронный ресурс] // Excode.ru [web-сайт]. – Режим доступа: http://www.excode.ru/art6072.html. – 19.12.2018.
3. POSIX [Электронный ресурс] // Википедия: свобод. энцикл. – Режим доступа: https://ru.wikipedia.org/wiki/POSIX. – 14.12.2018.
4. Федорчук, А. Введение в POSIX'ивизм [Электронный ресурс] / А. Федорчук. – Режим доступа: http://www.linuxcenter.ru/lib/books/posixbook/. – 14.12.2018.
5. Слинкин Д. А. Управление роботами на базе контролеров Lego Ev3 / Д. А. Слинкин. 2017-2018, 81 с.
6. MY PRACTIC [Электронный ресурс] // ПИД регулятор. Принцип действия, математическое описание, настройка. – Режим доступа: http://mypractic.ru/urok-40-pid-regulyator-princip-dejstviya-matematicheskoe-opisanie-nastrojka.html/. – 15.12.2018.
7. Овсяницкая, Л.Ю. Пропорциональное управление роботом Lego Mindstorms EV3 / Л.Ю. Овсяницкая, Д.Н. Овсяницкий, А.Д. Овсяницкий. – М.: Издательство «Перо», 2015. – 188 с.
8. Долинский, М. С. Алгоритмизация и программирование на Turbo Pascal / М. С. Долинский. – М.: Издательство «Питер», 2004 – 235 с.

Размещено на http://www.allbest.ru/

9. Алексеев, Е. Р. Самоучитель по программированию на Free Pascal и Lazarus / Е. Р. Алексеев, О. В. Чеснокова, Т. В. Кучер. – М.: Издательство «ДонНТУ УНИТЕХ», 2011 – 503 с.

10. Карпов, В. Э. ПИД – управление в нестрогом изложении / В. Э. Карпов. – М.: Издательство «Москва», 2012 – 35 с.

11. Скороспешкин В.Н., Скороспешкин М.В. Модифицированный псевдолинейный ПИД- регулятор. // Науковедение. – 2013. - № 6.

12. Юревич Е. И. Основы робототехники. - 2-е изд., перераб. и доп. - СПб.: БХВ-Петербург, 2005. - 416 с.