Управление двигателем с помощью Arduino: Пример информационного моделирования и разработки системы управления

Pravallika Vinnakota, MathWorks

Настройка регуляторов на физическом прототипе, или целевом оборудовании, может быть весьма рискованной, и повлечь за собой даже повреждение оборудования. Более надежным является метод построения модели целевой системы и симуляция её работы. Этот метод позволяет без лишнего риска убедиться в работоспособности регулятора при различных режимах эксплуатации задолго до запуска на реальном оборудовании.

В случае, если изначально создать модель целевой системы не представляется возможным - альтернативой является создание модели на основе экспериментальных данных, полученных в ходе экспериментов с прототипом устройства. Упрощенная линейная модель системы может быть достаточной для проектирования на начальных стадиях. Но детальный анализ и разработка высокоточного регулятора предъявляет высокие требования к точности целевой модели, которая зачастую является нелинейной.

В этой статье, на примере простой системы управления для двигателя постоянного тока, буде описан процесс идентификации модели целевой системы на основе базовой модели и экспериментальных данных, а также создания регулятора для полученной системы. Рабочий процесс состоит из следующих этапов: сбор данных, выявление линейных и нелинейных частей в целевой системе, разработка и моделирование регулятора с обратной связью, реализации регулятора на микропроцессоре для тестирования в режиме реального времени.

Готовые модели и исходный код, используемые в данном примере, доступны для скачивания.

Двигатель постоянного тока: Постановка задачи

На физическом уровне система состоит из двигателя постоянного тока, подключенного к плате Arduino® Uno через драйвер двигателя (рис. 1). Нашей целью является разработка регулятора с обратной связью для позиционирования ротора двигателя в определенном положении. Задача регулятора - выдавать соответствующее напряжение на выходе, основываясь на данных о текущем положении ротора двигателя.

В зависимости от напряжения на клеммах будет изменяться крутящий момент на валу двигателя, что в свою очередь повлечет за собой изменение скорости вращения. Для измерения угла поворота вала двигателя будет использован потенциометр, данные с которого мы подадим на вход микропроцессора с регулятором.

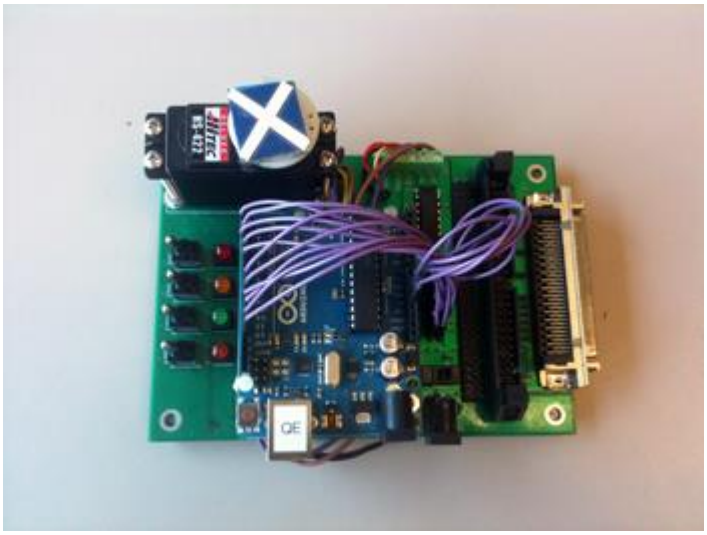

Рисунок 1. Плата Arduino соединена с двигателем постоянного тока.

Плата драйвера двигателя необходима для обеспечения необходимого тока в обмотках и дает возможность управлять двигателем в обоих направлениях. Данные о положении вала двигателя с потенциометра подаются на аналоговый вход платы Arduino. Имея эти данные, мы можем легко вычислить ошибку между фактическим положением вала и требуемым. Задание управляющего напряжения выполняется с помощью двух выходов с широтно-импульсной модуляцией (ШИМ). ШИМ поступает на вход драйвера, который управляет величиной тока в обмотках нашего двигателя.

Задача регулятора - поддерживать систему стабильной и обеспечивать быстрое реагирование на управляющий сигнал с минимальной установившейся ошибкой и перерегулированием.

Проведение измерений и обработка данных

К плате Arduino мы можем подключиться с помощью персонального компьютера (ПК) и встроенной поддержки Arduino в Simulink®. Мы можем прямо из модели генерировать исполняемый файл и запускать его на выбранном оборудовании. На рисунке 2 отображена библиотека блоков Simulink для использования совместно с Arduino.

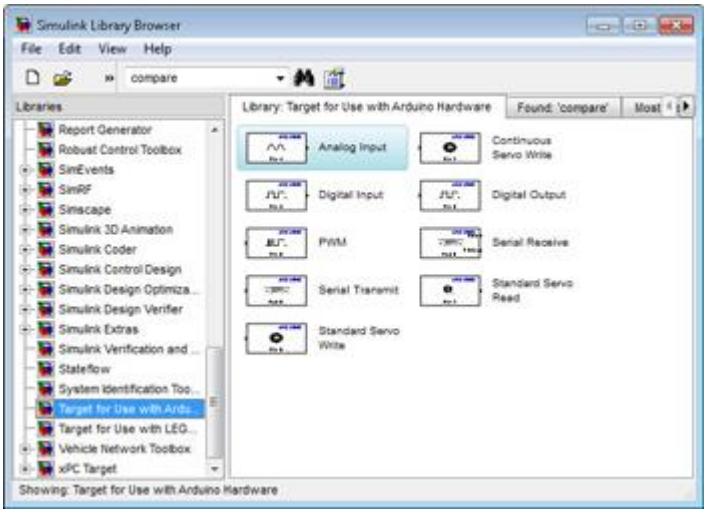

Рисунок 2. Библиотека блоков Simulink для использования с Arduino.

Для начала мы будем просто подавать напряжение на двигатель и измерять угол поворота вала. Для этого необходимо создать модель в Simulink, которая будет отвечать за сбор данных. Генерировать заданное напряжение и получать данные о положении вала будем с ПК, для чего необходимо обеспечить двухстороннюю связь между ним и платой Arduino. Для этого мы создаем вторую моделью, делающую возможным такую связь.

В модели, которая будет работать на плате Arduino Uno (рис. 3), блок "Voltage Command To Pins" читает приходящие на последовательный порт команды и управляет подачей ШИМ на соответствующие контакты макетной платы. Для связи ПК с платой Arduino мы используем COM порт. В подсистеме "CreateMessage" информация о позиции вала двигателя, полученные от одного из аналоговых входных контактов на плате, преобразуются в пакет данных для передачи через последовательный порт на компьютер.

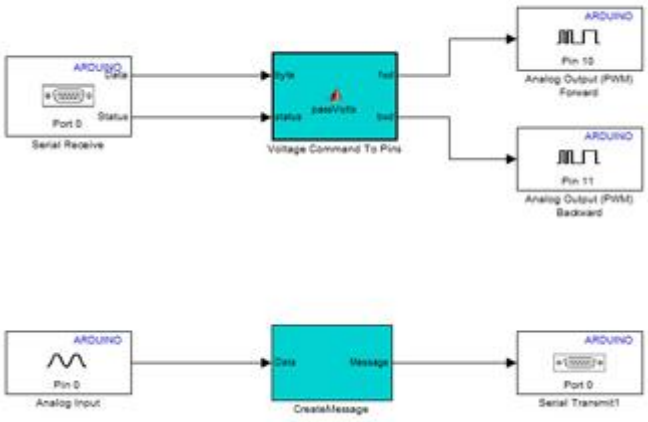

Рисунок 3. Модель Simulink, которая будет работать на Arduino.

Запуск приложения в реальном времени производится выбором Tools > Run on Target Hardware > Run. Вот и все, мы готовы получать данные об отклике системы с помощью модели, которая работает на компьютере (рис. 4).

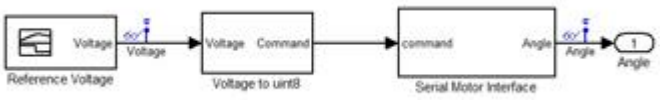

Рисунок 4. Модель, которая будет работать на ПК.

Эксперимент состоит в подаче различных уровней напряжения на двигатель и снятии положения вала. В конце моделирования мы получим пакет данных в рабочем пространстве MATLAB в виде объектов time-series.

Следующий этап - подготовка собранных данных для анализа. Используя следующий скрипт, мы преобразуем данные в объект iddata для импорта в System Identification Tool из System Identification Toolbox™.

```
>> Logsout
      logsout =
        Simulink.SimulationData.Dataset
        Package: Simulink.SimulationData
        Characteristics:
                    Name: 'logsout'
          Total Elements: 2
        Elements:
          1: 'Voltage'
          2: 'Angle'
        -Use getElement to access elements by index or name.
        -Use addElement or setElement to add or modify elements.
        Methods, Superclasses

>> u = logsout.getElement(1).Values.Data;

>> y = logsout.getElement(2).Values.Data;

>> bounds1 = iddata(y,u,0.01,'InputName','Voltage','OutputName','Angle',...
             ...'InputUnit','V','OutputUnit','deg')
             Time domain data set with 1001 samples.
             Sample time: 0.01 seconds
             Outputs       Unit (if specified)
                Angle         deg
             Inputs        Unit (if specified)
                Voltage       V
```

Далее для работы мы будем использовать 12 наборов данных. Эти данные были выбраны для создания разностороннего воздействия на систему. Они также позволят проверить работоспособность модели после её создания.

Разработка модели на основе экспериментальных данных

Разработка модели целевой системы с использованием методов системной идентификации - это всегда компромисс между точностью полученной модели и затраченными на её создание усилиями. Чем точнее модель мы хотим получить, тем выше будут затраты времени и труда. Наша цель состоит в том, чтобы создать наиболее простую модель, которая при этом будет надлежащим образом отражать динамику работы системы.

Типичный процесс системной идентификации начинается с пробы применения упрощенной линейной модели. После мы можем добавить учет нелинейности и увеличить общую сложность модели двигателя. Линейная модель может быть вполне применима для большинства случаев управления, но исследование нелинейности позволяет нам намного более точно симулировать реальное поведение системы и разрабатывать регулятор для работы в широком диапазоне рабочих точек.

Идентификация линейной системы

Используя объекты iddata, сперва рассмотрим передаточную функцию в непрерывном времени, как шаблон линейной динамической модели нашей целевой системы. Для этого нам необходимо указать количество полюсов и нулей функции. System Identification Toolbox автоматически определит их местоположение и максимально возможно приблизит отклик переходной функции к отклику, полученному от системы в ходе эксперимента.

Для запуска System Identification Tool необходимо набрать в командной строке следующую команду:

```
>> ident
```

Теперь мы можем импортировать наборы данных из рабочей области с помощью выпадающего меню Import Data (рис. 5). У нас также есть возможность предварительной обработки импортированных данных. Перед стартом процесса нам необходимо выбрать данных, которые будут использоваться отдельно для расчета параметров функции и для последующей валидации этих параметров после их нахождения. Один и тот же набор данных может быть использован как для расчета параметров, так и для их проверки. После завершения расчета мы можем также использовать другие наборы данных для дополнительной проверки. На рисунке 5 показан инструмент System Identification Tool с импортированными данными. Набор для первого расчета, data set 11, получен в сравнительно спокойном режиме работы и не содержит в себе явных нелинейностей.

Рисунок 5. System identification Tool с импортированными данными.

Теперь мы можем определить передаточную функцию на основе этих данных. В нашем примере мы возьмем передаточную функцию в непрерывном времени с двумя полюсами без нулей. (рис. 6).

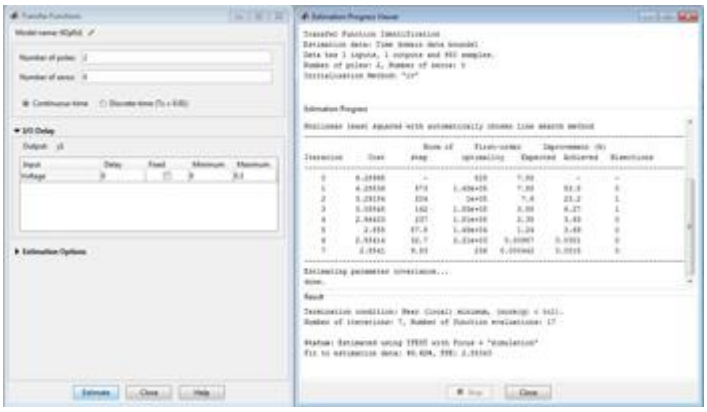

Рисунок 6. Непрерывная передаточная функция.

В процессе расчета мы сравниваем отклик расчетной модели и измеренные данные. Совпадение отклика расчетной линейной модели и экспериментальных данных составляет 93,62% (рис. 7).

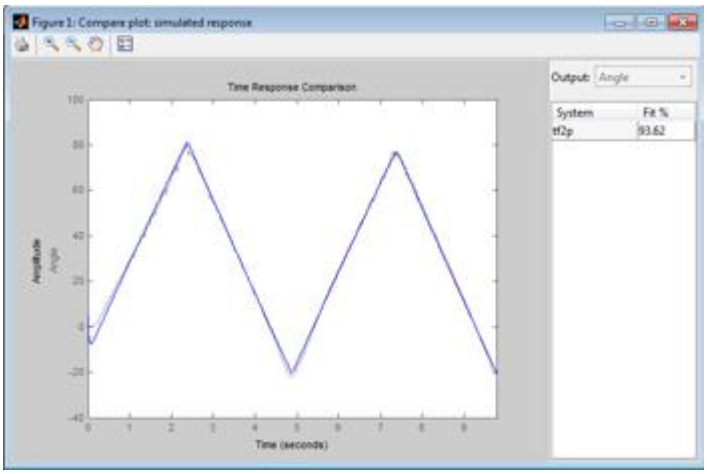

Рисунок 7. Сравнение отклика расчетной модели и экспериментальных данных.

Для оценки работы полученной передаточной функции в динамическом режиме мы должны проверить её с использованием независимого набора данных, который не

участвовал в процессе расчета параметров. Для этого мы выбираем набор данных 12, при котором двигатель также ведет себя линейно. Как видно из рисунка 8, мы получили довольно точную модель.

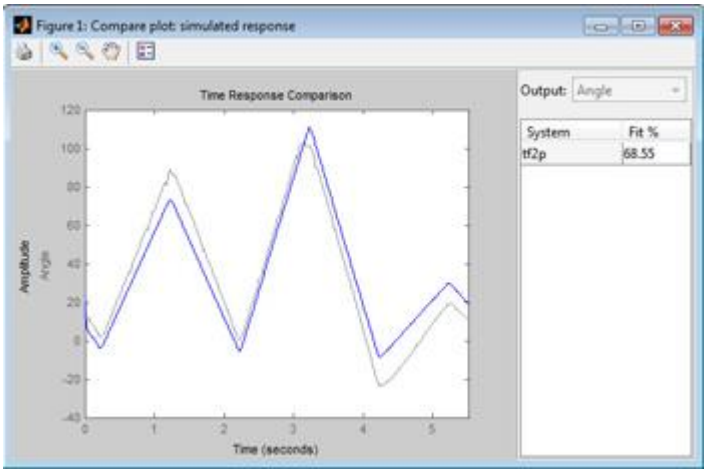

Рисунок 8: Проверка расчетной модели с использованием независимого набора данных.

Совпадение отклика модели не идеально, но при этом линейная модель довольно хорошо описывает динамику системы. Полученная передаточная функция вполне может использоваться нами далее в процессе разработки регулятора.

В добавок, мы также имеем возможность осуществить вероятностный анализ параметров целевой системы. Модель, полученная с помощью System Identification Toolbox содержит в себе информацию не только о номинальных значениях параметров функции, но и позволяет оценить распределение вероятности этих параметров в виде ковариационной матрицы. Мы можем получить оценку надежность модели, расчетную вероятность влияния внешних возмущений на систему, наличие не учтенной динамики. Распределение вероятности может быть отображено амплитудно-частотной характеристике (АЧХ). На рисунке 9 отображена АЧХ полученной передаточной функции и доверительный интервал вокруг значения характеристики при номинальных параметрах.

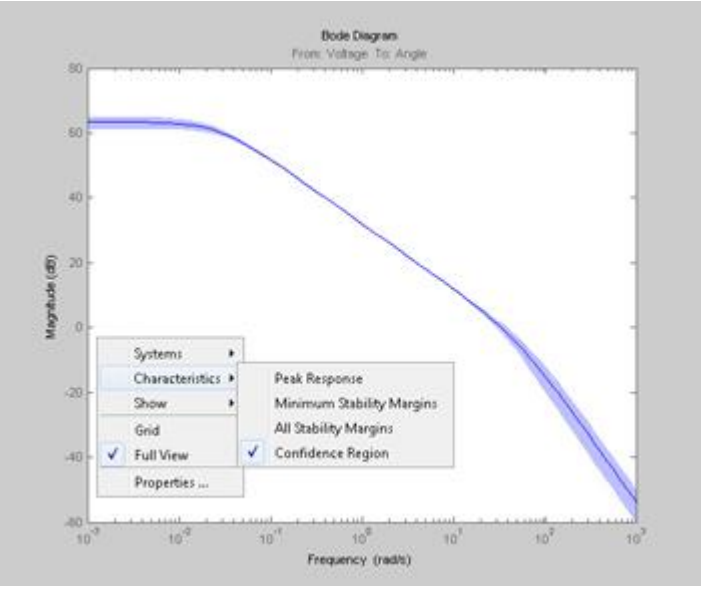

Рисунок 9 - АЧХ передаточной функции и её доверительный интервал.

Идентификация нелинейной системы

Линейная модель динамики двигателя, созданная на основе данных, собранных на линейных участках, является несомненно полезной для разработки. Однако эта модель не описывает нелинейное поведение, которое имеет место при более широком диапазоне режимов работы двигателя. Например, из набора данных 2 мы видим, что наш двигатель имеет ограничение поворота ротора в районе угла около 100°. А при рассмотрении набора данных 3 видно, что двигатель не восприимчив к слабым управляющим сигналам, возможно, из-за сухого трения.

На этом этапе мы постараемся создать более точную модель двигателя постоянного тока. Чтобы сделать это, мы будем работать с нелинейным представлением модели. При более пристальном рассмотрении экспериментальных данных мы уже определились с тем, что изменение угла поворота ротора двигателя имеет явно нелинейную связь с поданным на него напряжением. Также присутствует проявление гистерезиса из-за сил трения. Шаблон нелинейной модели ARX (NLARX) значительно более гибок, и может быть использован для описания такого поведения, используя богатый набор нелинейных функций, таких как вейвлеты (wavelets) и сигмоидальные нейронные сети. Эти функции позволяют использовать наши знания и предположения о присутствующих в системе нелинейностях в виде пользовательской настройки регрессии.

Для эффективного использования NLARX мы должны использовать данные содержащие достаточно информации о нелинейном поведении. Мы объединим три набора данных для создания полного набора. Остальные пять наборов будут использоваться как один большой набор для проверки.

```
>> mergedD = merge(z7,z3,z6)
      Time domain data set containing 3 experiments.
      Experiment    Samples      Sample Time
         Exp1         5480            0.01
         Exp2          980            0.01
         Exp3          980            0.01

      Outputs       Unit (if specified)
         Angle         deg

      Inputs        Unit (if specified)
         Voltage       V
>> mergedV = merge(z1,z2,z4,z5,z8);
```

Нелинейная модель имеет широкий спектр компонентов для настройки. Мы можем настраивать порядок модели, вносимую временную задержку, тип нелинейной функции, структуру и количество составных частей. В первую очередь добавим компоненты, которые представляют насыщение и поведение запретной зоны. После нескольких итераций, мы выбрали структуру модели, которая состоит из параллельно соединенных нейронной сети и линейной функции, а также ряд последовательно соединенных регрессоров, используемых для вычисления входа для предыдущей сборки. На основе этой структуры будут рассчитаны параметры для достижения максимального соответствия отклика с экспериментальными данными (рис. 10).

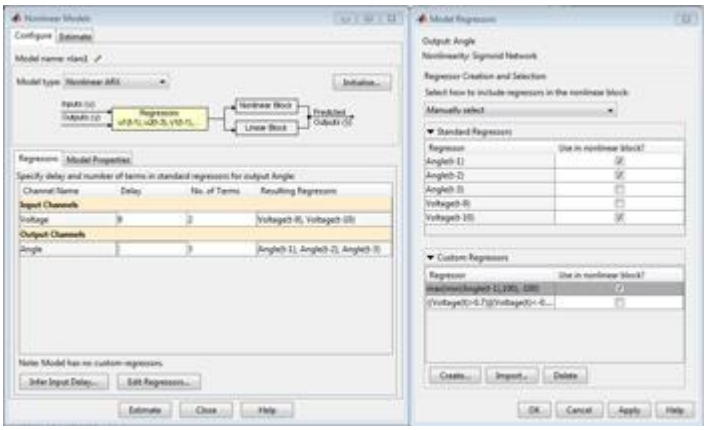

Рисунок 10 - Нелинейная ARX модель.

Отклик полученной модели имеет отличное совпадение (> 90%) для данных, которые использовались для расчета параметров, а также для проверочных данных. Эта модель может быть использована для синтеза регуляторов, анализа поведения системы и прогнозирования.

Синтез регулятора

Теперь мы готовы начать разработку точного ПИД-регулятора для нелинейной модели. Для этого достаточно выбрать рабочую точку, линеаризовать модель в её окрестности, а затем разработать контроллер для этой линеаризованной модели.

На рисунке 11 показана настройка параметров ПИД-регулятора по заданным требованиям.

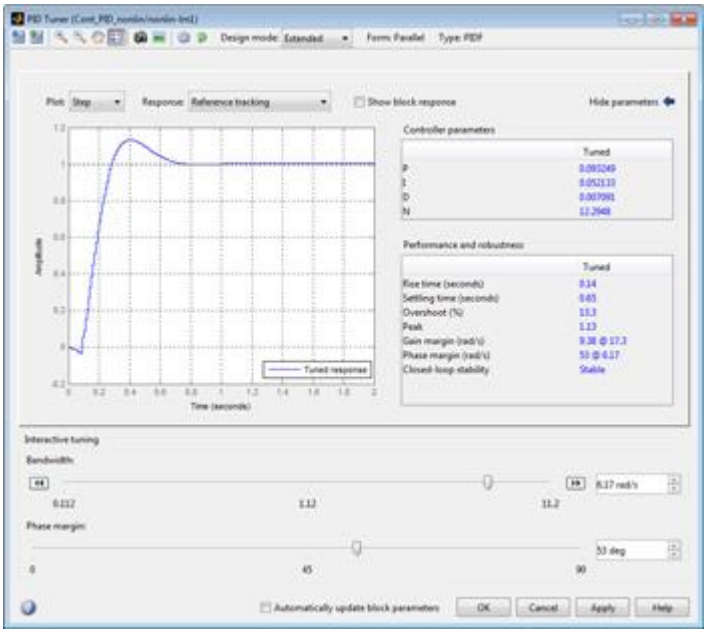

Рисунок 11 - Пользовательский интерфейс PID Tuner

После настройки мы готовы к проверке регулятора на нелинейной модели. На рисунке 12 показана модель Simulink для получения отклика от нелинейной ARX модели.

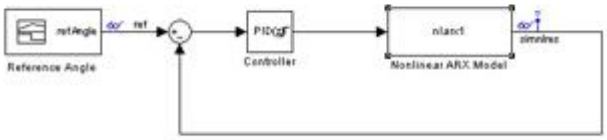

Рисунок 12 - Simulink модель для тестирования регулятора на нелинейной модели.

После запуска модели мы можем сравнить отклик нелинейной модели с обратной связью и линеаризованной модели для заданной позиции ротора 60° (рис. 13).

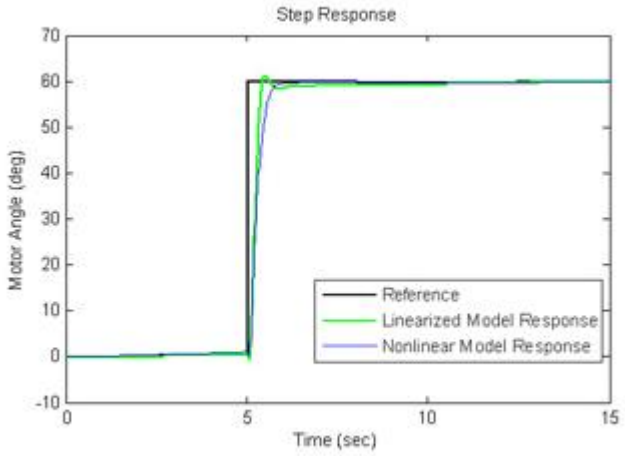

Рисунок 13. Сравнение откликов нелинейной и линеаризованной модели на ступенчатое воздействие.

Тестирование регулятора на целевом микропроцессоре

Для тестирования на целевом процессоре мы создаем модель Simulink с регулятором и загружаем её в Arduino Uno (рис. 14).

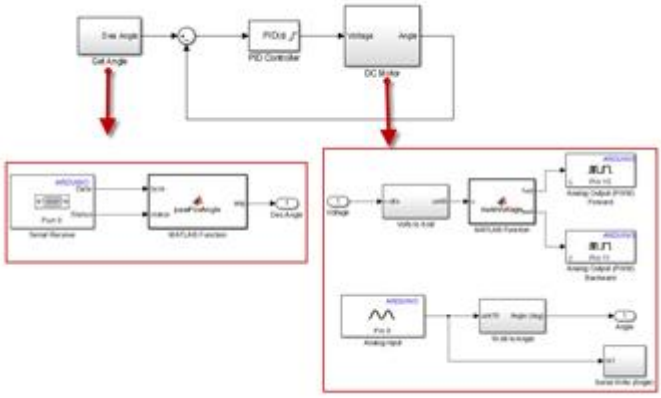

Рис. 14. Модель с регулятором для реализации на Arduino. Подсистема Get Angle получает управляющий сигнал через последовательный порт и обеспечивает необходимый угол поворота ротора двигателя. Подсистема DC Motor служит для настройки взаимодействия Arduino реальным двигателем.

Мы разработали регулятор линеаризовав нелинейную ARX модель в необходимой рабочей точке. И как результат, реальное поведение системы очень близко к результатам моделирования (рис. 15).

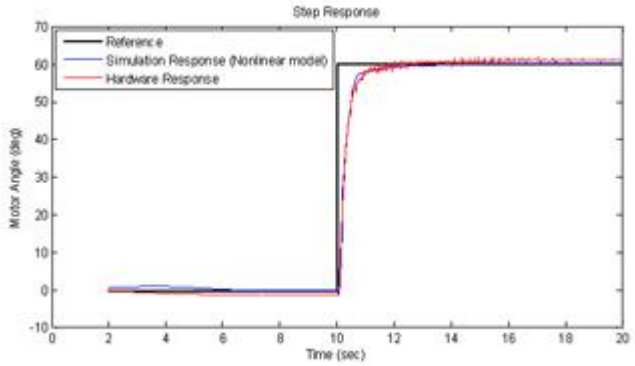

Рисунок 15. Сравнение моделирования и данных эксперимента на ступенчатое управляющее воздействие.

Мы также проверили, насколько хорошо регулятор отслеживает случайный управляющий сигнал. Аппаратные характеристики отслеживания также очень близки к полученным в ходе моделирования (рис. 16).

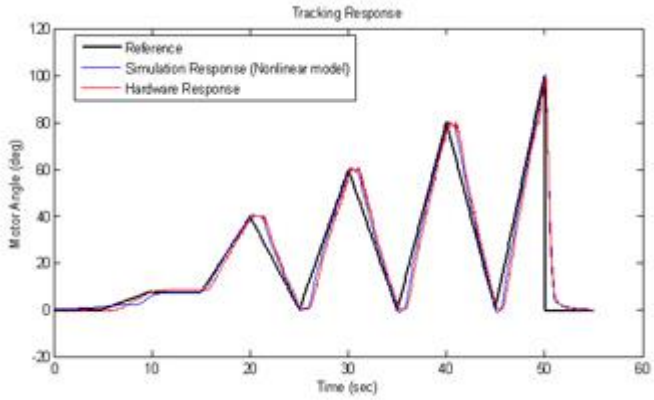

Рисунок 16. Сравнение откликов нелинейной модели и реального устройства.

Представленный пример, несмотря на свою простоту, охватывает все основные шаги разработки систем управления на основе экспериментальных данных. В процессе работы мы собрали экспериментальные данные с целевого устройства, и с помощью System Identification Toolbox создали нелинейную модель.

Мы показали, как можно создавать модели низкой и высокой достоверности, и синтезировать регуляторы используя их. Затем мы проверили работоспособность созданного регулятора на имеющейся в наличии макетной плате Arduino.

Использованные продукты

- MATLAB
- Simulink
- Simulink Control Design
- System Identification Toolbox