УЧЕБНАЯ МАСТЕРСКАЯ

СДЕЛАЙ САМ

Яркоев Константин Евгеньевич

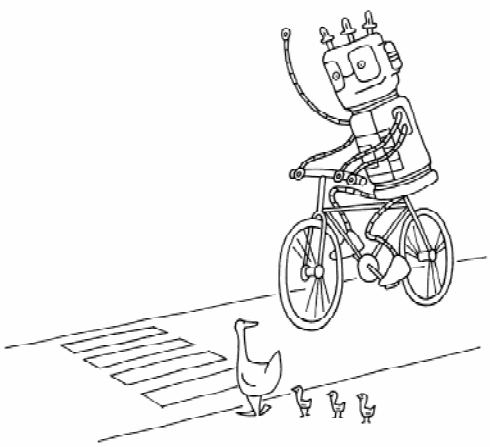
ПРАКТИЧЕСКАЯ МИКРОЭЛЕКТРОНИКА. ЗАНЯТИЕ 5. НОСИМАЯ ЭЛЕКТРОНИКА. ВЕЛОСИПЕДНАЯ СПЕЦОДЕЖДА

В этой статье мы сделаем проект носимой электроники – велосипедную спецодежду – с помощью носимой Arduino-совместимой платы Lilypad.

В наше время идет активная популяризация велосипедного вида транспорта. Возле крупных гипермаркетов появляются велопарковки, в городах по всему миру появляются отдельные велодорожки, а велотуризм привлекает все больше молодых людей. Массовость данного вида транспорта, а также доступность многих компонентов носимой электроники повлияли на тему данной статьи. В специализированных интернет-магазинах каждый может приобрести светодиоды, токопроводящую нить и относительно новую, но успевшую завоевать популярность во всем мире программно-аппаратную платформу Arduino. С помощью велокуртки можно повысить безопасность велосипедистов во время маневров на дороге.

Всем известно, что для подачи сигналов поворота и торможения водитель транспортного средства должен пользоваться световыми указателями поворота, а при их отсутствии либо неисправности – сигналами рукой (п. 8.1, 8.2 ПДД). Сигналу торможения соответствует вытянутая вверх рука. В то же время, велосипедисту запрещено ездить, не держась за руль хотя бы одной рукой (п. 24.3 ПДД), да и просто резко вскинуть руку в сторону весьма опасно! Торможение с вытяну-

Рис. 1. Велокуртка



той вверх левой (правой) рукой чревато резким поворотом влево (вправо) с последующим падением. Существующие технологии носимой электроники позволяют уже сегодня решить противоречие в ПДД и сделать велосипедное движение менее опасным. Используя кнопки и светодиоды белого цвета, можно реализовать поворотники. С помощью иголки и токопроводящей нити можно вшить электронную схему прямо в одежду (рис. 1).

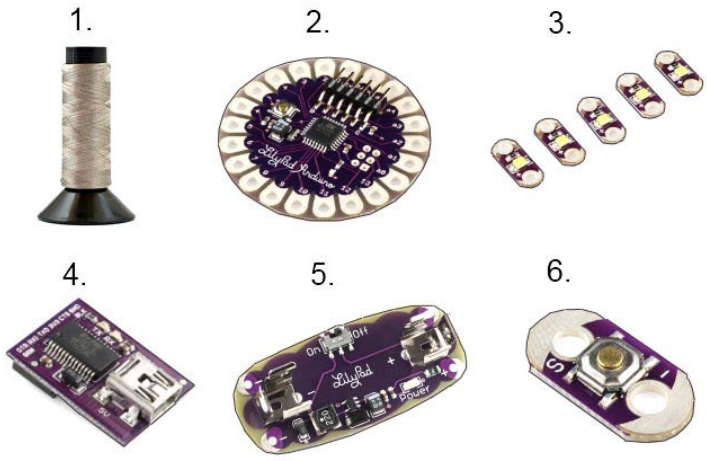


Рис. 2. Основные компоненты спецодежды

ПРОЕКТ 5. ВЕЛОСИПЕДНЫЙ КОСТЮМ

«Предполетная подготовка»

Для проекта нам потребуются следующие компоненты, купленные в интернет-магазине http://www.sparkfun.com (рис. 2):

1. Токопроводящая нить с серебряным покрытием. Сопротивлением 45 Ом/метр (1 катушка).
2. Arduino-совместимая плата Lilypad (1 шт.).
3. Пакет светодиодов желтого или белого цвета для повортоников; пакет светодиодов красного света для мерцающего режима в темное время суток (опционально).
4. Переходник USB-RS232 для прошивки Lilypad (1 шт.).
5. Блок питания для батарейки типоразмера AAA с выключателем (1 шт.).
6. Кнопки (2 шт.).
7. Иголка (1 шт.).
8. Ножницы (1 шт.).
9. Мел или мыло для выкройки
10. Батарейка AAA

Все компоненты стираемы: нить с серебряным напылением, контакты на светодиодах, кнопках, микроконтроллере Lilypad позолочены. МК может прошиваться (программироваться) с помощью преобразователя RS-232 в USB. В данном случае преобразователь вынесен в отдельное устройство для уменьшения размеров и веса платы Lilypad. Принцип действия устройства: микроконт-

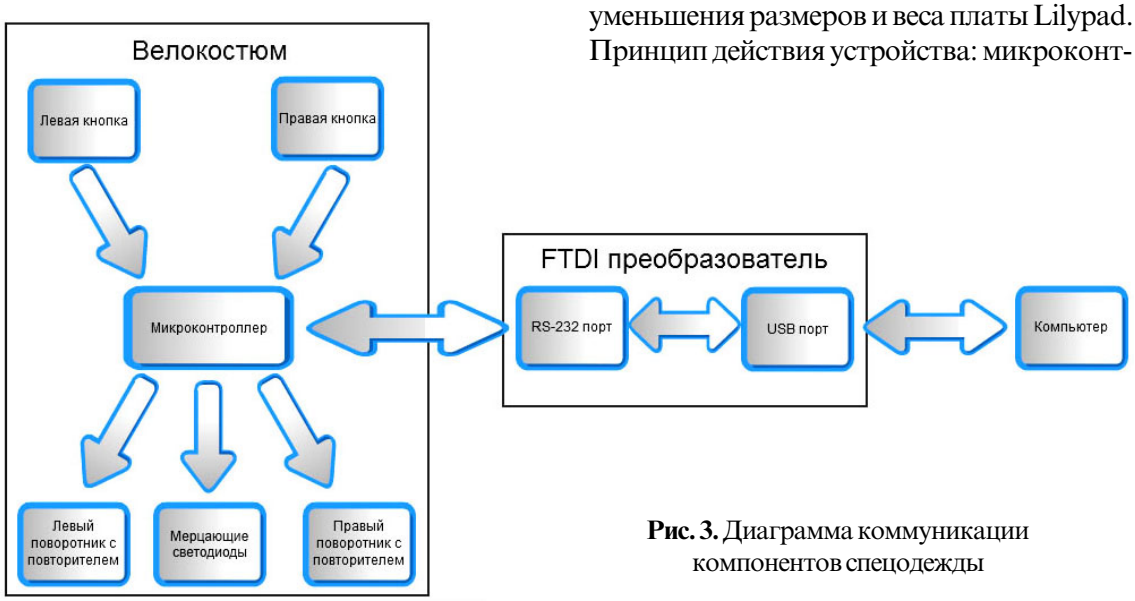


Рис. 3. Диаграмма коммуникации компонентов спецодежды

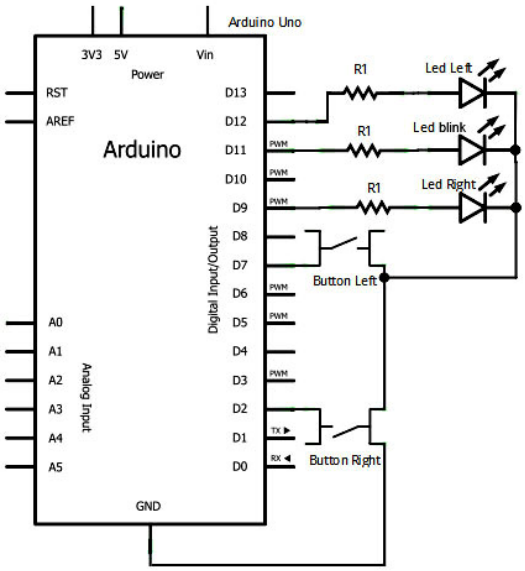


Рис. 4. Принципиальная схема

роллер (МК) опрашивает кнопки указателей поворота. При нажатии велосипедистом на кнопку поворотника соответствующие указатели поворота и повторитель указателя поворота должны мигать с заданной частотой. При повторном нажатии на эту же кнопку мигание прекращается. Если велосипедист включил правый поворотник, забыв при этом выключить левый, то МК должен автоматически выключить левый поворотник и включить правый (и наоборот) (рис. 3).

«Полет»

Для тестирования и отладки можно использовать следующий прототип принципиальной схемы спецодежды (рис. 4).

Ключи Button Left и Button Right подают на цифровые входы D7 и D2 значение 0 («Земля»), когда они замкнуты.

МК подает напряжение на цифровые выходы D13 и D9, к которым подключены светодиоды Led Left и Led Right.

МК подает напряжение на цифровой выход D11, к которому подключен светодиод Led Blink для постоянного мерцания (опционально).

Проблема дребезга кнопки

Существует общая проблема всех схем, точнее цифровых схем, включающих в себя механическую кнопку, – дребезг контактов кнопки.

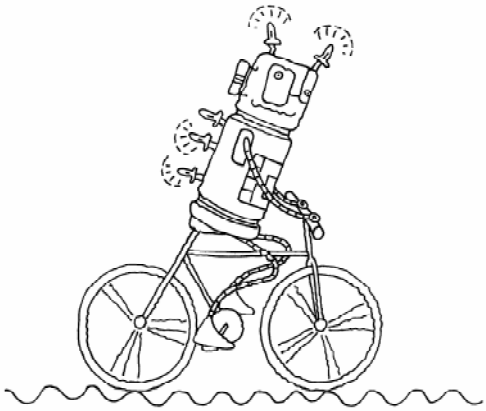
Дребезг контактов – это явление многократного неконтролируемого замыкания и размыкания контактов в моменты их соприкосновения или расхождения. Это явление приводит к формированию серии импульсов вместо требуемого одиночного перепада напряжения (рис. 5)

Из-за дребезга может возникнуть многократное непредсказуемое срабатывание логики цифрового устройства. Причины возникновения дребезга:

1. Кнопка – механическое устройство, содержащее в себе пружину, которая в момент замыкания (размыкания) начинает «пружинить» (колебаться) и не позволяет надежно зафиксировать контакт.

2. Сам контакт покрыт тончайшим слоем оксида, не проводящим электричество. В момент контакта возникает электрическая дуга (искрение) и за короткий промежуток времени происходит спекание контактов. Чем больше ток, тем быстрее происходит спекание.

Дребезг длится в течение нескольких миллисекунд, время варьируется в зависимости от степени износа кнопки, точнее, от износа пружины и стирания позолоты с пятна контакта. В некоторых приложениях дребезг некритичен, но в нашей программе требуется однозначно определить момент нажатия на кнопку, чтобы запустить процедуру мигания единожды. Кроме того, надо исключить ложные срабатывания, которые могут возникнуть из-за помехи с соседнего вывода микроконтроллера, либо из-за виб-



рации контакта при движении по неровной дороге.

С точки зрения программы, дребезг – это многократные переключения логических состояний, которые должны быть отфильтрованы и не приняты во внимание. Фильтрация осуществляется, исходя из того, что время дребезга для каждой кнопки может быть определено апостериори (но не превышает 30 миллисекунд). Самое простое решение:

1) определить замыкание ключа,

2) подождать 10–30 миллисекунд,

3) если ключ до сих пор замкнут, то замыкание действительно имеет место; иначе – игнорировать его.

Аналогично можно определить размыкание. Однако если МК будет просто ждать и ничего не делать, то мы можем пропустить срабатывание другой кнопки, датчика (например, датчика освещенности или датчика ускорения). Кроме того, кнопка со временем «стареет», контакты изнашиваются, время дребезга увеличивается. Поэтому придется все время эксплуатации переписывать программу и перепрошивать микроконтроллер, учитывая данный фактор.

Можно взять адаптивный алгоритм, основанный на самокалибровке. Он использует тот факт, что период колебаний τ намного меньше времени T (рис. 5) и составляет от 3 до 5 миллисекунд. Мы будем отслеживать время с момента последнего колебания, а также считать количество переключений состояний (ВКЛ-ВЫКЛ и ВЫКЛ-ВКЛ). Если после последнего переключения состояния прошло достаточно времени и количество переключений – четное число, то переключения состояния не произошло (поймали помеху). Если количество переключений – нечетное число, то произошло переключение состояния.

Инициализация:

1. счетчик нажатий = 0;

предыдущее состояние = ключ открыт;

t = текущее время;

Цикл:

2. определить текущее состояние ключа (закрыт или открыт) и текущее время;

3. если состояние ключа отличается от предыдущего, увеличиваем счетчик нажатий на 1, t = текущее время;

4. если разница между текущим временем и t больше порогового значения и счетчик нажатий – нечетное число, то состояние кнопки действительно изменилось;

5. предыдущее состояние = текущее.

Здесь знак равенства означает присвоение левой части значения правой части.

Алгоритм работы программы

Поворотники могут находиться в трех различных состояниях: ВЫКЛЮЧЕНЫ (оба поворотника должны быть выключены), ЛЕВЫЙ (должен мигать левый поворотник и левый повторитель), ПРАВЫЙ (должен

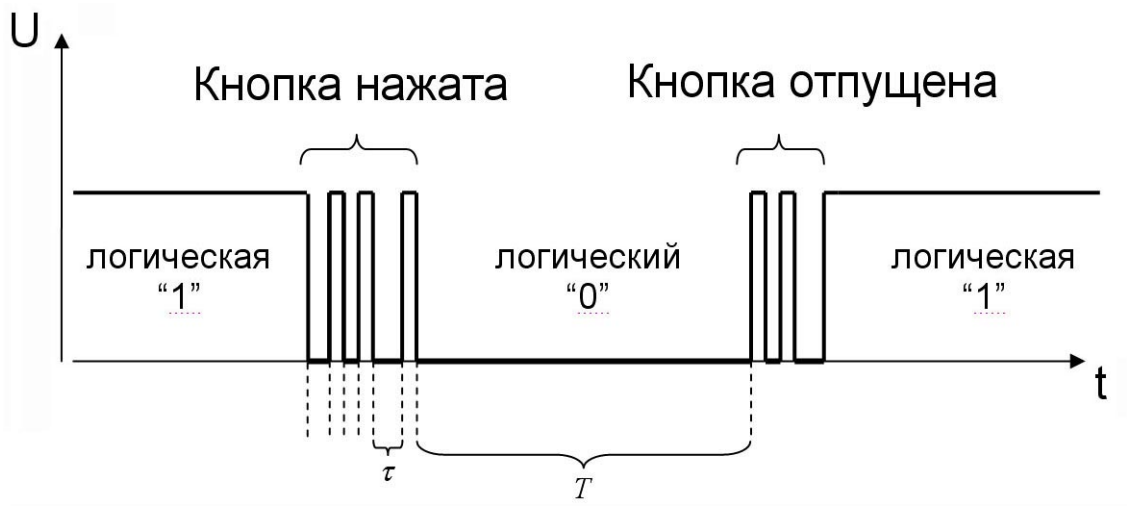


Рис. 5. Дребезг контактов

мигать правый поворотник и правый повторитель)

Цикл

1. Опрашиваем левую и правую кнопки с помощью адаптивного алгоритма определения дребезга кнопки.

2. Если нажата левая кнопка и состояние ВЫКЛЮЧЕНЫ, то перейти в состояние ЛЕВЫЙ.

3. Если нажата левая кнопка и состояние ЛЕВЫЙ, то перейти в состояние ВЫКЛЮЧЕНЫ.

4. Если нажата левая кнопка и состояние ПРАВЫЙ, то перейти в состояние ЛЕВЫЙ.

5. Если нажата правая кнопка и состояние ВЫКЛЮЧЕНЫ, то перейти в состояние ПРАВЫЙ.

6. Если нажата правая кнопка и состояние ПРАВЫЙ, то перейти в состояние ВЫКЛЮЧЕНЫ.

7. Если нажата правая кнопка и состояние ЛЕВЫЙ, то перейти в состояние ПРАВЫЙ.

8. Помигать левым или правым поворотником (или не мигать) в зависимости от текущего состояния.

Сборка прототипа на беспаечной макетной плате показана на рис. 6.

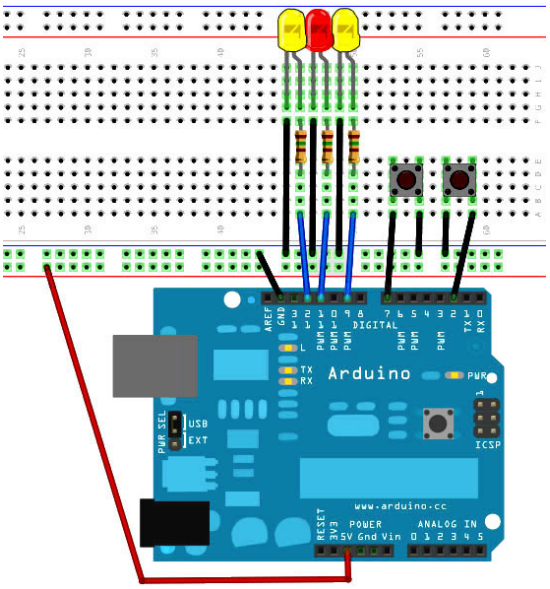


Рис. 6. Тестирование на макетной плате

Шитье

Для данной работы вам понадобится готовая куртка. Так как токопроводящая нитка весьма толстая, необходимо выбирать материал куртки достаточной толщины. Перед шитьем нарисуйте выкройку мелом (как показано на рис. 8, 9, 10). Нужно шить так, чтобы стежки не были видны снаружи. Старайтесь шить по пройме рукава (рис. 7). Поскольку сопротивление нитки зависит от ее длины, нужно стараться минимизировать длину ниток. В районе манжет необходимо шить зигзагом, чтобы нитка не порвалась при растягивании манжет.

Если нитка порвалась, свяжите место разрыва и капните немного клея на узел для прочности. После окончания шитья и завершения проверки работоспособности необходимо покрыть видимую часть швов лаком для предотвращения короткого замыкания.

Код

На основе описанного алгоритма составлена программа для прошивки костюма в среде Arduino 1.0. Код с комментариями смотрите в приложении к журналу на диске.

Проект состоит из 5 основных файлов:
Bicycle.ino,
DebounceButton.h,
DebounceButton.ino,
TurnSignalState.h,
TurnSignalState.ino.

DebounceButton.h, DebounceButton.ino содержат описание и реализацию класса кнопки с фильтрацией дребезга.

TurnSignalState.h, TurnSignalState.ino содержат описание и реализацию класса, отвечающего за мигание поворотников, в зависимости от состояния.

Файл Bicycle.ino – самый главный файл, содержит основной цикл и номера выводов.

Рис. 7. Шов идет по пройме рукава

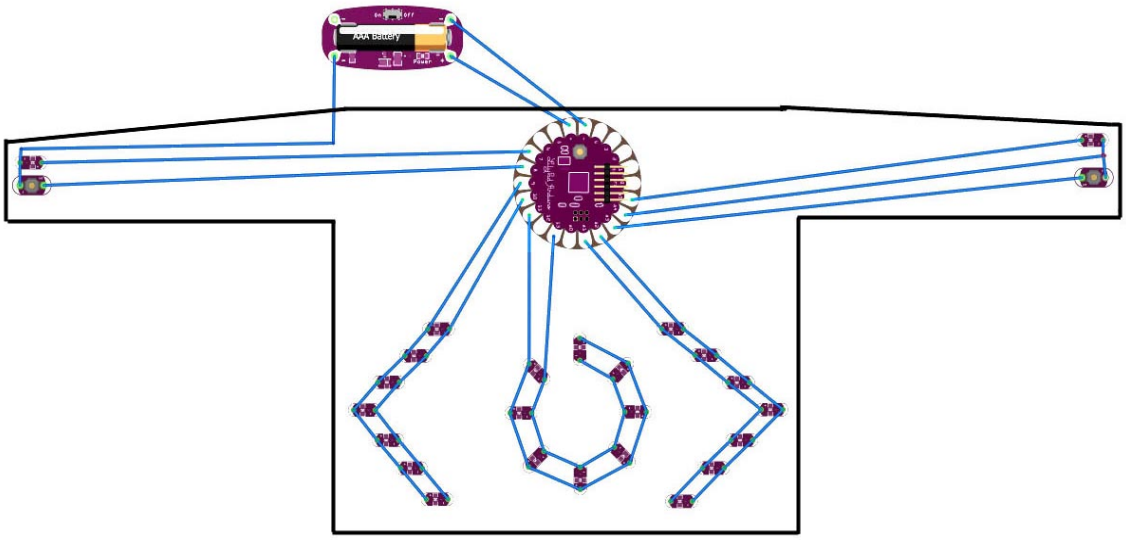


Рис. 8. Пример выкройки. Блок питания пришивается снаружи на груди

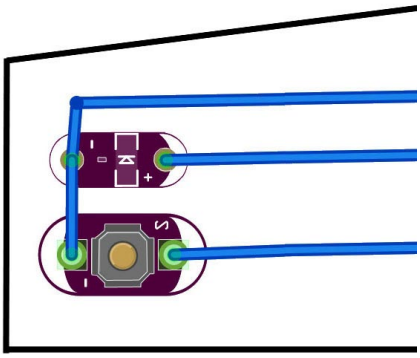


Рис. 9. Пример выкройки на манжете

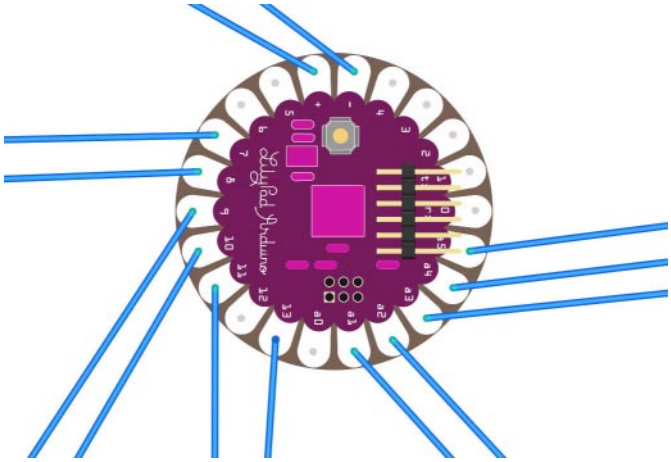


Рис. 10. Показаны задействованные разъемы

Яркоев Константин Евгеньевич,
разработчик фирмы SoftDev SPb.