Dr. Bob Davidov

Создание интерактивных объектов и сред на базе платформы Arduino

Цель работы: рассмотрение вариантов сопряжения сред проектирования МатЛАБ, Simulink, LabVIEW с внешней средой через электронную платформу (контроллер) Arduino.

Задача работы: подключение аппаратного интерфейса к среде проектирования MatLAB.

Приборы и принадлежности: Персональный компьютер, контроллер Arduino UNO или DFRduino UNO R3, USB кабель, датчик температуры LM35, Интегрированная среда разработки Arduino -1.0.4., COM Port Toolkit, MATLAB R2008a или новее, Simulink - MATLAB R2010a или новее, LabView.

ОБЩИЕ СВЕДЕНИЯ

Arduino — аппаратная вычислительная платформа (универсальный контроллер), основными компонентам которой являются плата ввода/вывода и среда разработки на языке Processing/Wiring. Через последовательный порт Arduino можно подключить и к другим программным средам, например, Matlab, Simulink, LabVIEW и они, таким образом, получают инструмент для взаимодействия с реальными объектами систем управления, модулями визуализации или робототехники.

MATLAB m-файлы или Simulink поддерживают совместную работу с контроллером Arduino Mega 2560 или Arduino Uno по USB каналу в режиме RS-232 последовательного соединения. Это взаимодействие основано на выполнении серверной программы контроллера, который принимает команды через последовательный порт, выполняет их и, при необходимости, возвращает результат. Такой подход помогает

- запускать программы сразу при помощи загрузчика контроллера, без дополнительных средств (программаторов),
- работать в среде MATLAB или Simulink для интерактивной разработки и отладки,
- разрабатывать программы ввода аналоговых и цифровых данных,
- управлять двигателями постоянного тока, серводвигателями, и шаговыми двигателями,
- выполнять контурное управление с частотой до 25 Гц (не в реальном времени).
- создавать реальные электронно-механические устройства.

Рис. 1. Периферия платформы ARDUINO (DFRduino) UNO R3. Цифровые выходы, поддерживающие работу в режиме ШИМ обозначены на плате волнистой линией ~. AREF вывод опорного напряжение АЦП (используется с функцией analogReference()).

В этой работе используется аналог Arduino UNO: DFRduino UNO R3. Аналог, как и прототип, имеет следующую спецификацию:

- Микроконтроллер ATmega 328, частота 16 MHz,
- Напряжение питания: 7-12 В (6-20 В - предел). Вход используется для подачи питания от внешнего источника (в отсутствие 5 В от разъема USB).
- Вывод питания 3.3В/50 mA
- 32 Кб флэш память (2Кб занято загрузчиком),
- 2Кб ОЗУ
- 1Кб EEPROM
- 6 аналоговых (0-5В, 10бит, 0.1мс) вводов и 14 цифровых вводов / выводов (до 40 мА) с 6 PWM (ШИМ) выходами (~490 Гц, 0 .. 255).
- Встроенные USB-COM (300, .., 115200 бод), SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK), и I2C: 4 (SDA) и 5 (SCL)каналы связи;
- 2 внешних прерывания

В микроконтроллер предварительно прошит загрузчик, поэтому внешний программатор не нужен. На каждой платформе присутствует линейный стабилизатор напряжения 5В.

На рынке доступны платы расширения для Arduino, известные как «shields».

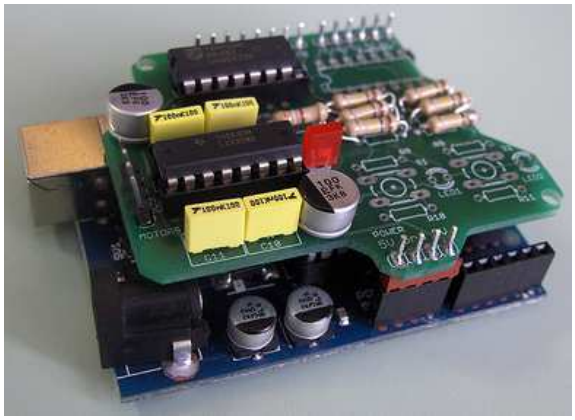


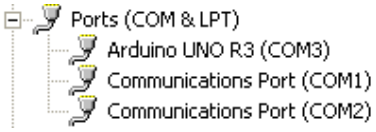
Рис. 2. Пример платы расширения «shields» для Arduino: модуль управления двигателем.

ПОДГОТОВКА К РАБОТЕ С ARDUINO ЧЕРЕЗ ИНТЕГРИРОВАННУЮ СРЕДУ РАЗРАБОТКИ (http://robocraft.ru/blog/arduino/98.html)

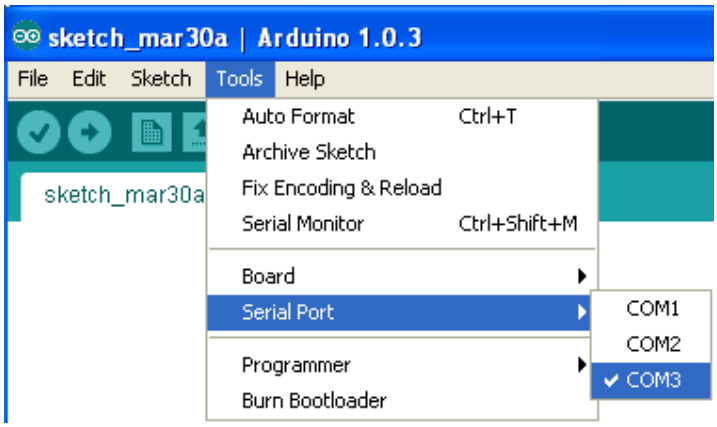
Для того, чтобы иметь возможность писать свои программы и записывать их на Arduino необходимо подключиться к Интегрированной среде разработки Arduino — это кроссплатформенное приложение на Java, включающее в себя редактор кода, компилятор и модуль передачи прошивки в плату. Среда разработки основана на языке программирования Processing. Строго говоря, это C/C++, дополненный некоторыми библиотеками. Программы обрабатываются с помощью препроцессора, а затем компилируется с помощью AVR-GCC.

Для подключения Adruino к интегрированной среде выполните следующие шаги.

1. Скачайте с сайта http://arduino.cc/en/main/software Arduino IDE (arduino-1.0.3-windows.zip).
2. Распакуйте zip файл и поместите директорию arduino-1.0.3 в коневой каталог диска C. Можно и в другое место, но главное, чтобы в пути не было названий отличных от английского.
3. Подключите плату к компьютеру, посредством USB-кабеля типа A-B.
4. Когда операционная система обнаружит новое устройство (ждать надо ~5 мин.) и предложит установить драйвер, укажите драйвер находящийся в директории arduino-1.0.3\drivers\. После установки в системе появится дополнительный COM-порт:



5. Запустите Arduino IDE –C:\arduino-1.0.3\arduino.exe.
6. Выберите новый COM-порт (Tools > Serial port)



7. Выберите тип платы: Tools > Board > Arduino UNO:

8. На этом закончена установка и подключение платформы Arduino, можно начинать программирование.

ПРОГРАММИРОВАНИЕ ARDUINO

Arduino/ Freeduino программируется на языке Wiring коды которого преобразуются с минимальными изменениями в программу на языке C/C++, и затем компилируются компилятором AVR-GCC. Так что, фактически, используется специализированный для микроконтроллеров AVR вариант C/C++.

Среда разработки, и набор базовых библиотек, упрощают доступ к периферии микроконтроллера.

Например, на языке Wiring установка скорости последовательного порта 9600 бит в секунду, задается всего одной строчкой: Serial.begin(9600);

Тогда как при использовании «голого» C/C++ пришлось бы разбираться с документацией на микроконтроллер, и вызывать нечто подобное:

```
UBRR0H = ((F_CPU / 16 + 9600 / 2) / 9600 - 1) >> 8;
UBRR0L = ((F_CPU / 16 + 9600 / 2) / 9600 - 1);
sbi(UCSR0B, RXEN0);
sbi(UCSR0B, TXEN0);
sbi(UCSR0B, RXCIE0);
```

фактически, класс HardwareSerial (\hardware\cores\arduino\HardwareSerial.h) инкапсулирует данные операции в функции begin (\hardware\cores\arduino\HardwareSerial.cpp)

Объявление COM-порта в самом заголовочном файле .h файле: extern HardwareSerial Serial. A

Подключение заголовочного файла класса работы с последовательным портом в программе происходит строчкой: #include "HardwareSerial.h"

Структура программы

В любом тестовом скрипте можно увидеть две необходимые функции: setup() и loop().

Функция setup() запускается один раз, после каждого включения питания или сброса платы Arduino. Она используется для инициализации переменных, установки режима работы цифровых портов, и т.д.

Функция loop() в бесконечном цикле исполняет команды, которые описаны в ее теле.

Рассмотрите простой пример:

```
void setup()        // начальные установки
{
  beginSerial(9600);  // установка скорости работы серийного порта на 9600 бит/сек
  pinMode(3, INPUT);  // установка 3-его порта на ввод данных
}

// Программа проверяет 3-ий порт на наличие на нём сигнала и посылает ответ в
// виде текстового сообщения на последовательный порт компьютера
void loop()                 // тело программы
{
  if (digitalRead(3) == HIGH)  // условие на опрос 3-го порта
    serialWrite('H');         // отправка сообщения в виде буквы «H» на COM-порт
  else
    serialWrite('L');         // отправка сообщения в виде буквы «L» на COM-порт
  delay(1000);                // задержка 1 сек.
}
```

Константы

Константы – предопределенные значения. Объявление констант (а так же базовых макросов и функций) можно увидеть в файле \hardware\cores\arduino\wiring.h

Уровни сигналов порта HIGH и LOW

```
#define HIGH 0x1          // 5V out, >3V in       “1”
#define LOW  0x0          // 0V out, <2V in       “0”
```

Таким образом, оба следующих вызова будут эквивалентны:

```
digitalWrite(13, HIGH);   // можно так,
digitalWrite(13, 1);      // а можно и так
```

Настройка цифровых портов на ввод (INPUT) и вывод (OUTPUT) сигналов

```
#define INPUT 0x0
#define OUTPUT 0x1
```

ВНИМАНИЕ:

- Порты поддерживают положительное или отрицательное направление тока до 40 мА

- Порты, сконфигурированные как выводы, могут быть повреждены, если их замкнуть накоротко на «землю» (общая шина питания), на источник питания +5 В, или подсоединить к мощной нагрузке с малым сопротивлением.

Пример:

```
pinMode(13, OUTPUT);  // 13-й вывод будет выходом
pinMode(12, INPUT);      // а 12-й – входом
```

Цифровой ввод/вывод

Функции Arduino для цифрового ввода-вывода объявлены в файле \hardware\cores\arduino\wiring.h, а реализованы в \hardware\cores\arduino\wiring_digital.c.

void pinMode(uint8_t, uint8_t);

Вызов: pinMode (порт, режим), где порт – значение целого типа от 0 до 13; режим – либо INPUT (ввод) либо OUTPUT (вывод).

Примечание: Аналоговые входы могут использоваться как цифровые входы/выходы, при обращении к ним по номерам с 14 по 19 (аналоговый вход 0 .. 5)

void digitalWrite(uint8_t, uint8_t);

Вызов: digitalWrite(порт, значение), где порт: номер порта; значение: HIGH или LOW.

Пример:

```
digitalWrite(13, HIGH);  // выставляем 13-й вывод в «высокое» состояние
```

int digitalRead(uint8_t);

Вызов: value = digitalRead (порт); Считывает значение на указанном порту, возвращает текущее значение на порту (HIGH или LOW) типа int

Пример:

```
int val;
val = digitalRead(12);  // опрашиваем 12-й вывод
```

Аналоговый ввод/вывод сигнала

int analogRead(uint8_t);

Вызов: value = analogRead(порт); Считывает значение с указанного аналогового порта. (Arduino содержит 6 каналов АЦП по 10 бит каждый. Входному напряжению от 0 - 5В соответствует возвращаемое число АЦП типа int от 0 до 1023). Требуется приблизительно 0.1 мс, чтобы считать значение аналогового ввода.

Пример:

```
int val;
val = analogRead(0);  // считываем значение на 0м аналоговом входе
```

Примечание: Аналоговые порты по умолчанию определенны на ввод сигнала и в отличие от цифровых портов их не требуется конфигурировать с помощью вызова функции pinMode.

void analogWrite(uint8_t, int);

Вызов: analogWrite(порт, значение); Выводит на порт аналоговое значение. Эта функция работает на: 3, 5, 6, 9, 10, и 11 цифровых портах Arduino в режиме ШИМ ~490 Гц в интервале 0 .. 255.

Пример:

```
analogWrite(9, 128);  // устанавливаем на 9 контакте значение ШИМ эквивалентное 2,5В
```

Примечание: Нет необходимости вызвать функцию pinMode, чтобы установить порт на вывод сигналов перед вызовом функции analogWrite.

Платформа Arduino имеет внутренний источник опорного напряжения (ИОН) для АЦП. Через резистор 5k к порту AREF можно подключить и внешний источник. Функция void analogReference(uint8_t mode); определяет источник опорного напряжения для АЦП.

Дополнительные функции ввода/вывода сигнала

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, byte val); Сдвиг байта данных по одному биту за раз.

unsigned long pulseIn(uint8_t pin, uint8_t state, unsigned long timeout); Считывает импульс (высокий или низкий) с цифрового порта и возвращает продолжительность импульса в микросекундах, импульс ожидается в течении таймаута (если не указать — будет ждать 1 сек.)

Работа со временем

unsigned long millis(void);

Вызов: time = millis(); Возвращает число миллисекунд, с момента исполнения Arduino текущей программы. Счетчик переполнится и обнулится приблизительно через 1193 часов или же 50 дней (49.7).

Пример:

```
unsigned long time; // объявление переменной time типа unsigned long
time = millis();    // передача количества миллисекунд
```

unsigned long micros(void); Возвращает число микросекунд, с момента исполнения Arduino/Freeduino текущей программы. Переменная переполнится (обнулится), приблизительно через 70 минут.

Примечание: На 16 МГц плате Arduino, данная функция работает с разрешением в 4 микросекунды.

Пример:

```
time = micros();
Serial.println(time);  // выводим число микросекунд с момента запуска программы
```

void delay(unsigned long);

Вызов: delay(время_мс); Приостанавливает программу на заданное число миллисекунд.

Пример:

```
delay(1000);  //пауза 1 секунда
```

void delayMicroseconds(unsigned int us);

Вызов: delayMicroseconds(время_мкс); Приостанавливает программу на заданное число микросекунд.

Математические функции

min(x, y) Возвращает меньшее из двух чисел

Пример:

```
sensVal = min(sensVal, 100); // sensVal не меняет значения, если он меньше 100
                             // получается, что sensVal не сможет превысить 100.
```

max(x, y) Возвращает большее из двух чисел

abs(x) Возвращает модуль (абсолютную величину) числа

constrain(x, a, b) Проверяет находится ли число x в диапазоне (a,b)
Возвращаемое значение: x: если x лежит между a и b
a: если x меньше, чем a
b: если x больше, чем b

map(value, fromLow, fromHigh, toLow, toHigh) Отображает число из одного диапазона в другой. Оперирует целыми числами

Пример:

```
y = map(x, 1, 50, 50, 1);
y = map(x, 1, 50, 50, -100);
/* отображает аналоговое значение к 8 битам (диапазон от 0 до 255) */
  int val = analogRead(0);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
```

pow(base, exponent) Функция предназначена для возведения числа в заданную степень.

sq(x) Вычисляет квадрат числа: число умноженное на себя.

sin(rad) Вычисляет синус угла (в радианах). Результат будет между -1 и 1.

cos(rad) Вычисляет косинус угла (в радианах). Результат будет между -1 и 1.

tan(rad) Вычисляет тангенс угла (в радианах). Результат будет между плюс и минус бесконечностью :)

Псевдослучайные числа

void randomSeed(unsigned int seed) инициализирует генератор псевдослучайных чисел

Пример:

```
long randNumber;
void setup(){
  Serial.begin(9600);
  randomSeed(analogRead(0));
}

void loop(){
  randNumber = random(300);
  Serial.println(randNumber);
  delay(50);
}
```

long random(long howbig)

long random(long howsmall, long howbig) Генерирует псевдослучайное число

Последовательная передача данных

Arduino имеет встроенный контроллер для последовательной передачи данных, который может использоваться как для связи между Arduino устройствами, так и для связи с компьютером. На компьютере соответствующее соединение представлено USB COM-портом, который появляется в системе после установки необходимого драйвера.

Связь происходит по цифровым портам 0 и 1, и поэтому Вы не сможете использовать их для цифрового ввода/вывода если используете функции последовательной передачи данных.

Serial.begin(long);

Вызов: Serial.begin(скорость_передачи); Устанавливает скорость передачи COM порта: 300, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 57600, или 115200 бит в секунду.

Serial.available(void);

Вызов: count = Serial.available(); Функция возвращает количество накопленных в буфере микроконтроллера байт принимаемых по последовательному порту. Последовательный буфер может хранить до 128 байт.

Пример:

```
if (Serial.available() > 0) { // Если в буфере есть данные
   // здесь должен быть прием и обработка данных
  }
```

Serial.read(void);

Вызов: char = Serial.read(); Считывает следующий байт из буфера последовательного порта. Возвращаемое значение: первый доступный байт входящих данных с последовательного порта, или -1 если нет входящих данных.

Serial.write(uint8_t c)

Вызов: Serial.write(val); // val: переменная для передачи, как единственный байт
Serial.write(str); // str: строка для передачи -последовательность байт
Serial.write(buf, len); Записывает данные в последовательный порт. Данные посылаются как байт или последовательность байт; для отправки символьной информации следует использовать функцию print(). buf: массив для передачи, как последовательность байт, len: длина массива

Serial.flush(void)

Вызов: Serial.flush(); Очищает входной буфер последовательного порта.

Serial.print() Вывод данных на последовательный порт в ASCII кодах.

Функция имеет несколько форм вызова в зависимости от типа и формата выводимых данных.

Пример:

Serial.print(b, format); b – число, format – формат выводимого числа:
DEC - десятичное представление числа b.
HEX - шестнадцатеричное представление числа b.
OCT - восьмеричное представление числа b.
BIN - двоичное представление числа b.
BYTE - выводит младший байт числа b.

Пример

```
int b = 79;
Serial.print(b, HEX); //выдаст в порт строку «4F»
```

Serial.print(str) если str – строка или массив символов, побайтно передает str на COM-порт.

Пример

```
char bytes[3] = {79, 80, 81};    //массив из 5 байт со значениями 79,80,81
Serial.print("Here our bytes:"); //выводит строку «Here our bytes:»
Serial.print(bytes);             //выводит 3 символа с кодами 79,80,81 –
                                 //это символы «OPQ»
```

Serial.print(b) если b имеет тип byte или char, выводит в порт само число b.

Serial.println() выводятся символ возврата каретки (ASCII 13, или '\r') и символ новой линии (ASCII 10, или '\n').

Пример

```
int b = 79;
Serial.print(b, DEC); //выдаст в порт строку «79»
Serial.print("\r\n"); //выведет символы "\r\n" – перевод строки
Serial.println(b, DEC); //выдаст в порт строку «79\r\n»
```

Прерывания

Прерывание (англ. interrupt) — сигнал, сообщающий процессору о наступлении какого-либо события. При этом выполнение текущей последовательности команд приостанавливается, и управление передаётся обработчику прерывания, который выполняет работу по обработке события и возвращает управление в прерванный код.

Функции Arduino для работы с прерываниями объявлены в файле \hardware\cores\arduino\wiring.h и реализованы в файле \hardware\cores\arduino\WInterrupts.c

void attachInterrupt(uint8_t, void (*)(void), int mode); Определяет, какую функцию вызывать, когда происходит внешнее прерывание. Замещает предыдущую функцию, если таковая была привязана к данному прерыванию. Большинство плат Arduino имеют два внешних прерывания с номерами 0 (на digital pin 2) и 1 (на digital pin 3). Arduino Mega имеет дополнительно ещё четыре: с номерами 2 (pin 21), 3 (pin 20), 4 (pin 19) и 5 (pin 18).

Вызов: attachInterrupt(interrupt, function, mode); где interrupt: номер прерывания (int); function: функция, которая должны вызываться при прерывании. Функция не должна принимать параметров и не должна ничего возвращать. mode: определяет, когда должно сработать прерывание:

LOW — вызов прерывания всякий раз, когда на порту низкий уровень напряжения;
CHANGE – прерывание вызывается при изменении значения на входе;
RISING – вызов прерывания при изменении уровня напряжения с низкого (LOW) на высокое(HIGH)
FALLING – вызов прерывания при изменении уровня напряжения с высокого (HIGH) на низкое (LOW)

Пример:

```
// светодиод, подключённый к digital pin 13 будет изменять своё
// состояние при изменении напряжения на digital pin 2
//
int pin = 13;
volatile int state = LOW; // переменная которая не оптимизируется компилятором
```

```
void setup()
{
  pinMode(pin, OUTPUT);              // порт как выход
  attachInterrupt(0, blink, CHANGE); // привязываем 0-е прерывание к функции
blink().
}

void loop()
{
  digitalWrite(pin, state);          // выводим state
}

void blink()
{
  state = !state;                    // меняем значение на противоположное
}
```

void detachInterrupt(uint8_t); Отключает указанное прерывание.

Вызов: detachInterrupt(interrupt); где interrupt: номер прерывания для отключения (0 или 1).

Энергонезависимая память EEPROM http://robocraft.ru/blog/arduino/82.html

EEPROM — (Electrically Erasable Programmable Read-Only Memory) электрически стираемое перепрограммируемое ПЗУ, ЭСППЗУ). Память такого типа может стираться и заполняться данными несколько десятков тысяч раз. Используется в твердотельных накопителях. Одной из разновидностей EEPROM является флеш-память (Flash Memory).

byte EEPROM.read(address) Считывает байт из энергонезависимой памяти EEPROM. Если байт до этого никогда не перезаписывался – вернёт значение 255. address: порядковый номер ячейки памяти для чтения — от 0 до 511 (int)

Пример (File-Examples-EEPROM-eeprom_read):

```
/*
  * Чтение EEPROM
  *
  * Считывает значения всех байтов энергонезависимой памяти
  * EEPROM и выводит их в COM-порт
  */

  #include <EEPROM.h>

  // начальный адрес памяти EEPROM
  int address = 0;
  byte value;

  void setup()
  {
    Serial.begin(9600);
  }

  void loop()
  {
```

```
    // считываем значение по текущему адресу EEPROM
    value = EEPROM.read(address);

    Serial.print(address);
    Serial.print("\t");
    Serial.print(value, DEC);
    Serial.println();

    // устанавливаем следующую ячейку памяти
    address = address + 1;

    // EEPROM содержит всего 512 байт: от 0 до 511, поэтому
    // если адрес достиг 512, то снова переходим на 0
    if (address == 512)
      address = 0;

    delay(500);
  }
```

void EEPROM.write(address, value) Записывает байт в энергонезависимую память. address: порядковый номер ячейки памяти для записи — от 0 до 511 (int); value: байт для записи – от 0 до 255 (byte)

Примечание: Документация (datasheet) на микроконтроллеры Atmega8/168 говорит, что возможное количество циклов перезаписи данных в памяти ограничено 100000 раз (Write/Erase Cycles). Время, требуемое для завершения цикла записи составляет 3.3 ms. Данная задержка уже учитывается библиотекой EEPROM, поэтому в дополнительном вызове delay() нет необходимости.

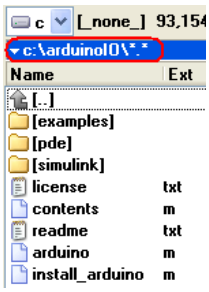
Очистка памяти производится записью нулей.

Создание своей библиотеки. http://robocraft.ru/blog/arduino/102.html

ПОДКЛЮЧЕНИЕ ARDUINO К MATLAB http://robocraft.ru/blog/741.html

Подключение Arduino к MATLAB (R2008a или более поздней версии) выполняйте в следующей последовательности:

1. Скачайте пакет ArduinoIO с официального сайта MathWorks: http://www.mathworks.com/matlabcentral/fileexchange/32374
2. Распакуйте пакет, например, в c:\arduinoIO:



3. Загрузите МатЛАБ

4. Выполните команды

   >>cd c:\arduinoIO % переход в директорию

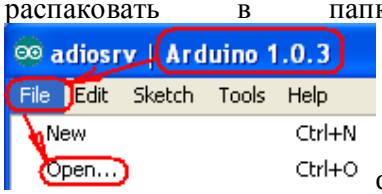
   >>install_arduino % установка

   Arduino folders added to the path

   Saved updated MATLAB path

   >>savepath

5. Запишите в платформу Arduino прошивку-ретранслятор. Ее задача принимать и выполнять команды из порта. Один вариант прошивки поддерживает только функции работы с аналоговыми и цифровыми портами, другой - еще и шаговые двигатели. Для последнего требуется специальная библиотека, которую можно взять в https://github.com/adafruit/Adafruit-Motor-Shield-library/zipball/master. Ее требуется распаковать в папку Arduino-XXX/libraries/ Открываем файл



   c:\arduinoIO\pde\adiosrv\adiosrv.pde или c:\arduinoIO\pde\srv\srv.pde из среды Arduino и прошиваем  .

6. В консоле MATLAB создайте объект: >> a=arduino('COM3')

7. Проверьте правильность подключения следующими командами обращения МатЛАБ к Arduino

   >>a.pinMode(13,'OUTPUT')
   >> a.digitalWrite(13,1) % запись в порт “1”, на плате должен загореться светодиод
   >> a.digitalWrite(13,0) % запись в порт “0”, диод на плате должен погаснуть
   >>foo=a.analogRead(1) % чтение с аналогового входа, при открытом входе foo > 0

ПОДКЛЮЧЕНИЕ ARDUINO К SIMULINK (http://robocraft.ru/blog/741.html)

Для подключения Arduino к Simulink (+SimulinkCoder + EmbeddedCoder) MATLAB (R2010a) или более поздней версии выполните следующие шаги:

1. Скачайте пакет SimulinkSupportPackageforArduino
   http://www.mathworks.com/matlabcentral/fileexchange/30277
2. Распакуйте пакет, например в c:\arduino_simulink
3. Загрузите MatLAB
4. Выполните следующие команды для добавления путей:

   >>cd c:\arduino_simulink
   >> addpath(fullfile(pwd,'arduino'),fullfile(pwd,'blocks'),fullfile(pwd,'demos'))
   >>savepath

5. Обновите кастомизацию:

   >>sl_refresh_customizations

6. Подключите платформу Arduino к USB порту компьютера .

7. Укажите путь к среде arduino

   >>arduino.Prefs.setArduinoPath('c:\ArduinoTarget')

8. Выведите список всех доступных платформ
   >> arduino.Prefs.setBoard

   Specify one of the following board labels:
   'uno' (Arduino Uno)
   'mega2560' (Arduino Mega 2560)
   'mega' (Arduino Mega (ATmega1280))

9. Задайте текущую платформу 'uno' командой

   >>arduino.Prefs.setBoard('uno')

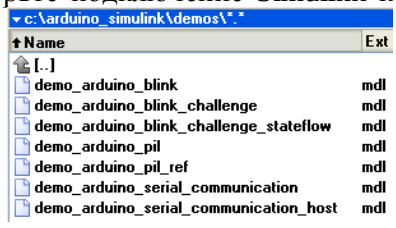
10. Проверьте доступные порты:

    >>comPorts=arduino.Prefs.searchForComPort

    В результате получите, например, comPorts ='COM3'

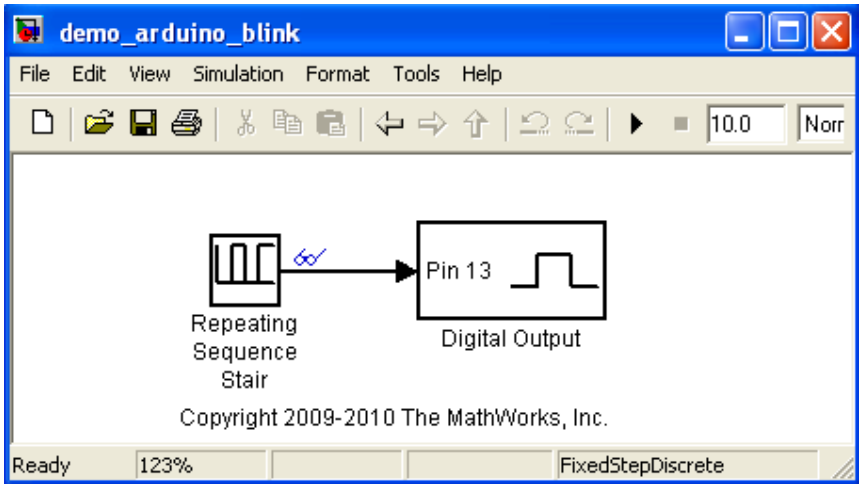
11. Подключите Arduino к COM порту:

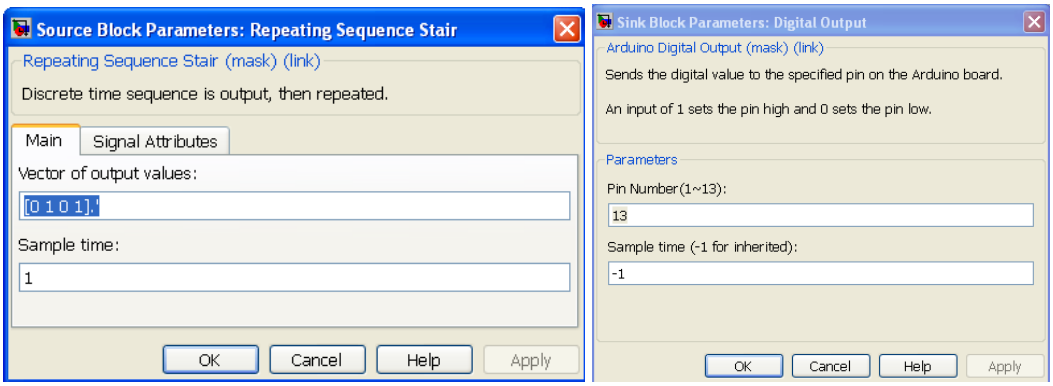
    >>arduino.Prefs.setComPort('COM3')

12. Проверьте подключение Simulink к Arduino, по работе демонстрационных примеров папки 

для этого:

    12.1 Откройте первый пример Simulink модели командой >>demo_arduino_blink и установите параметры модели:

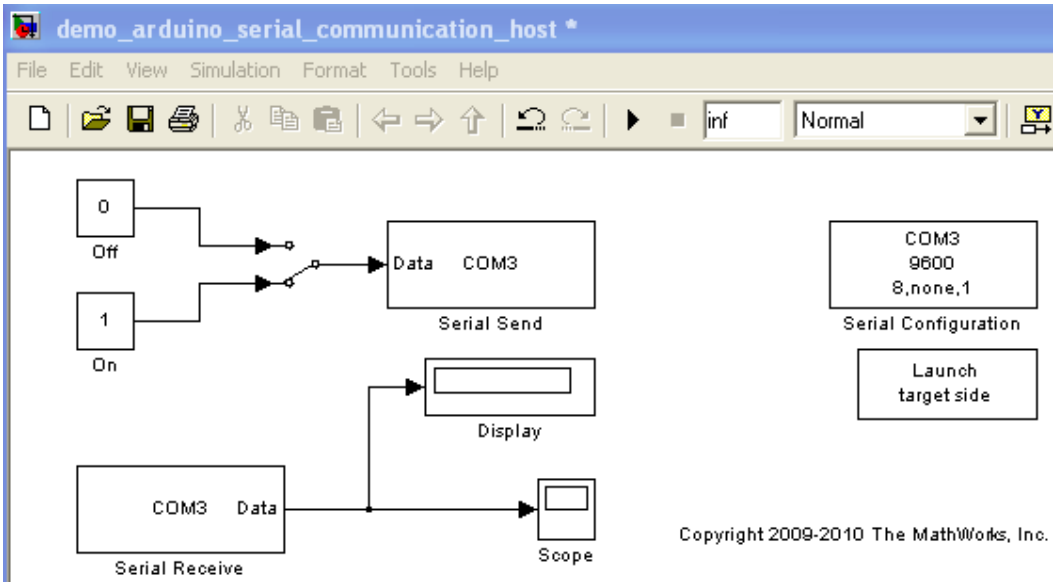




12.2 Комбинацией Ctrl+B или командой Tools>>Codegeneration>>BuildModel запустите кросс-компилятор.

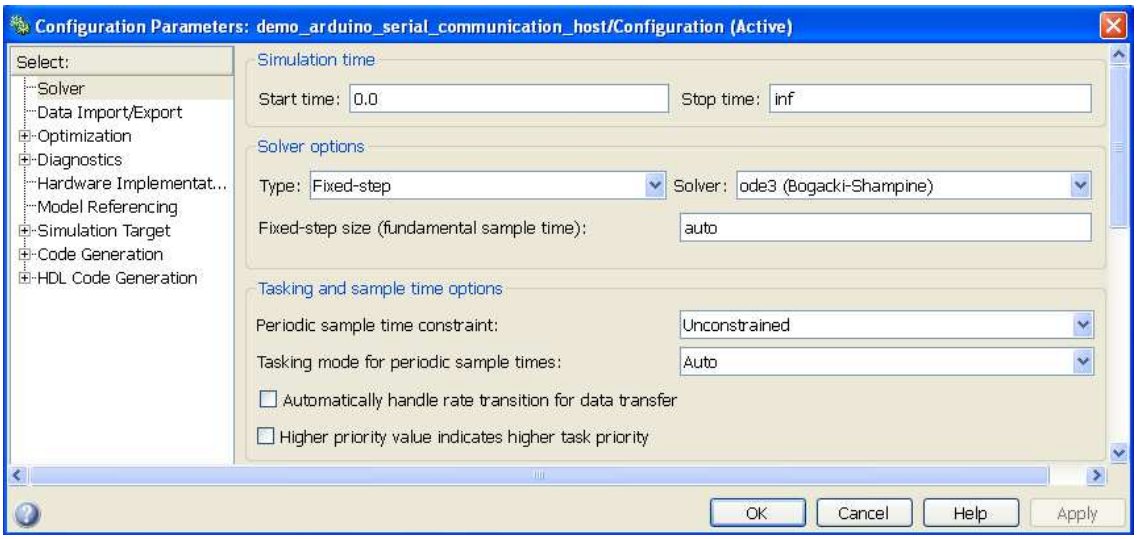
12.3 Нажмите кнопку “Start simulation”  код платформы прошивается и выполняется в Arduino. Контрольный светодиод Arduino (pin 13) начинает мигать.

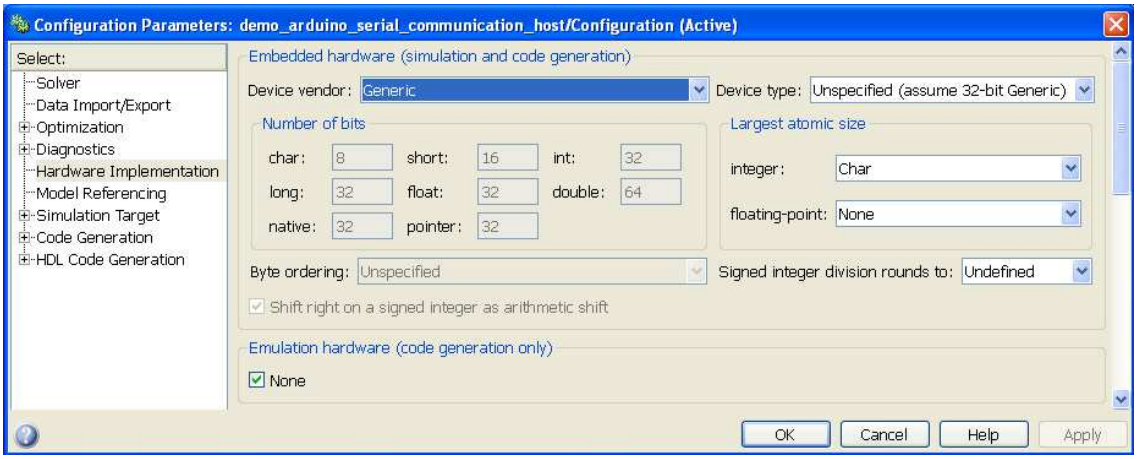
12.4 Для ручного управления светодиодом Arduino через Simulink и наблюдением за входом Pin 2 АЦП Arduino откройте другую модель demo_arduino_serial_communication_host.mdl:

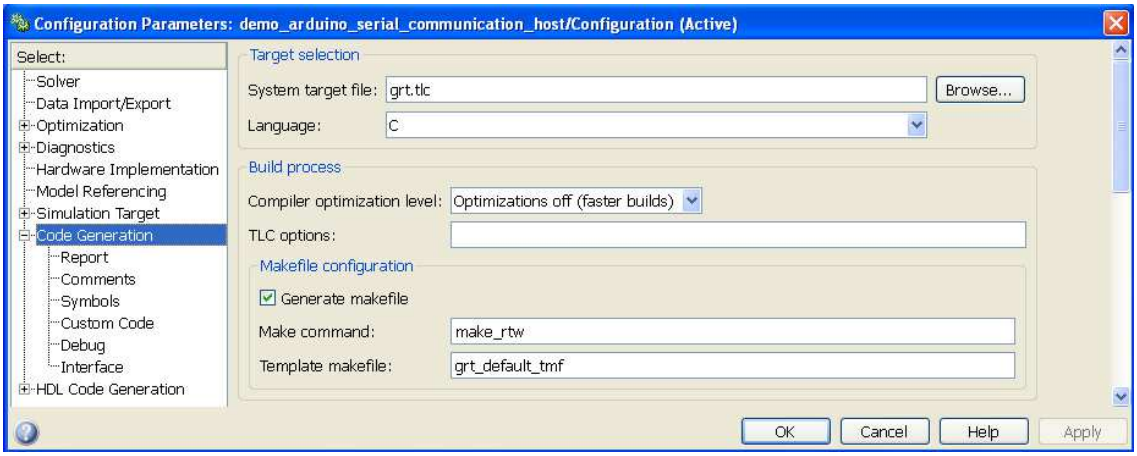


В блоке “Lunch target side” этой модели Simulink содержится код Arduino который загружается в него перед запуском модели.

Модель Simulink имеет следующие параметры.

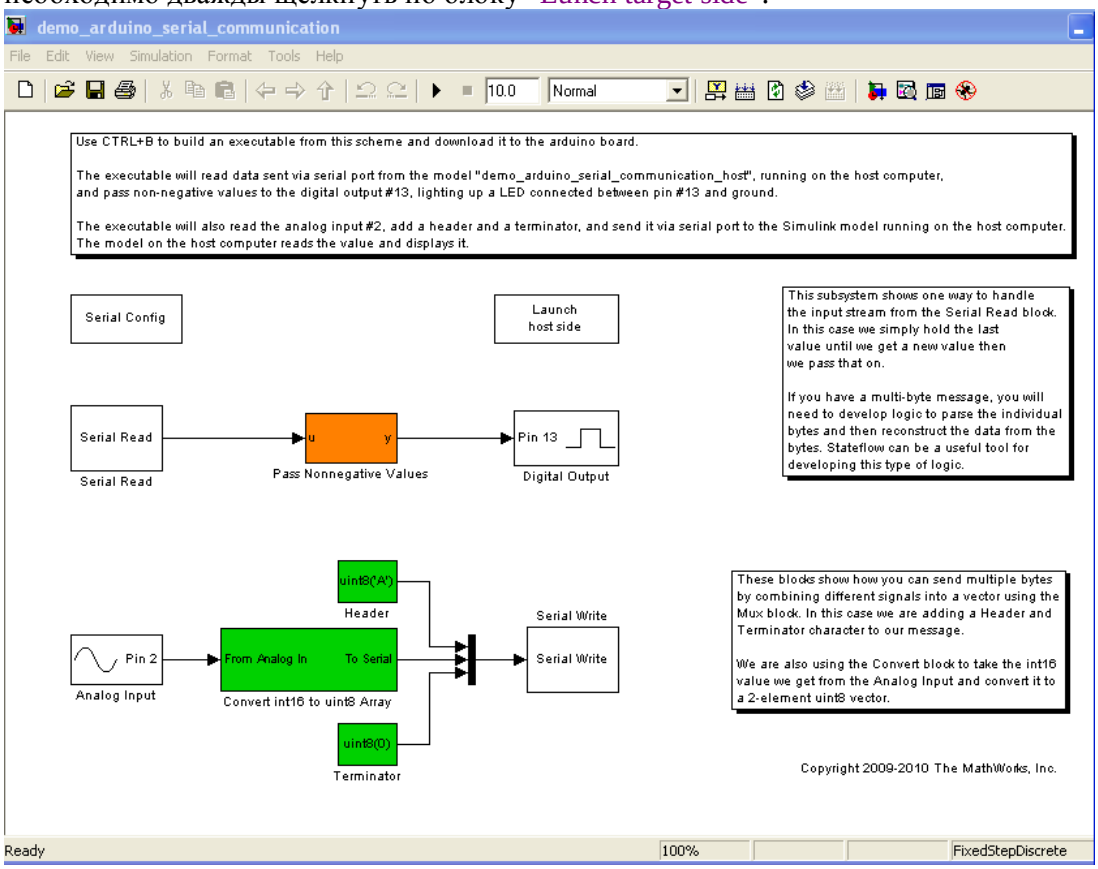






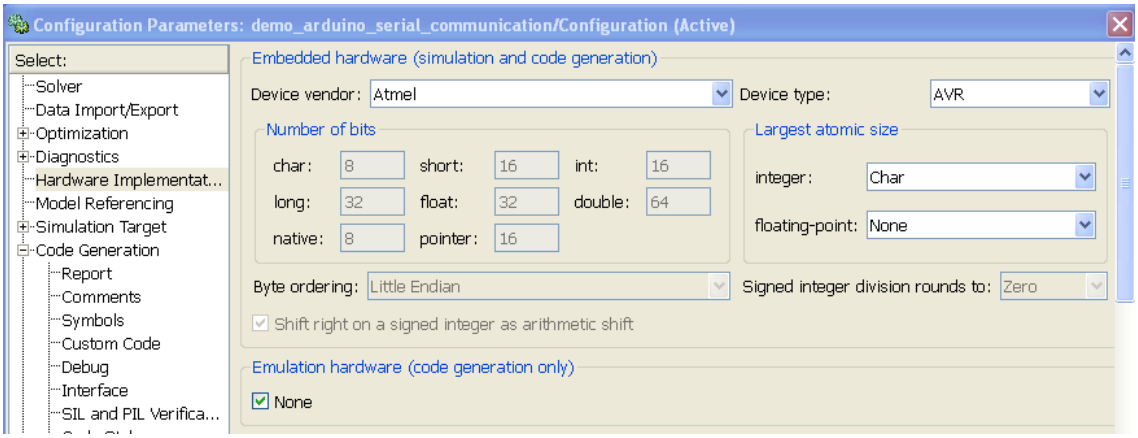
12.5 Настройте номера COM порта модели на номер COM порта платформы Arduino.

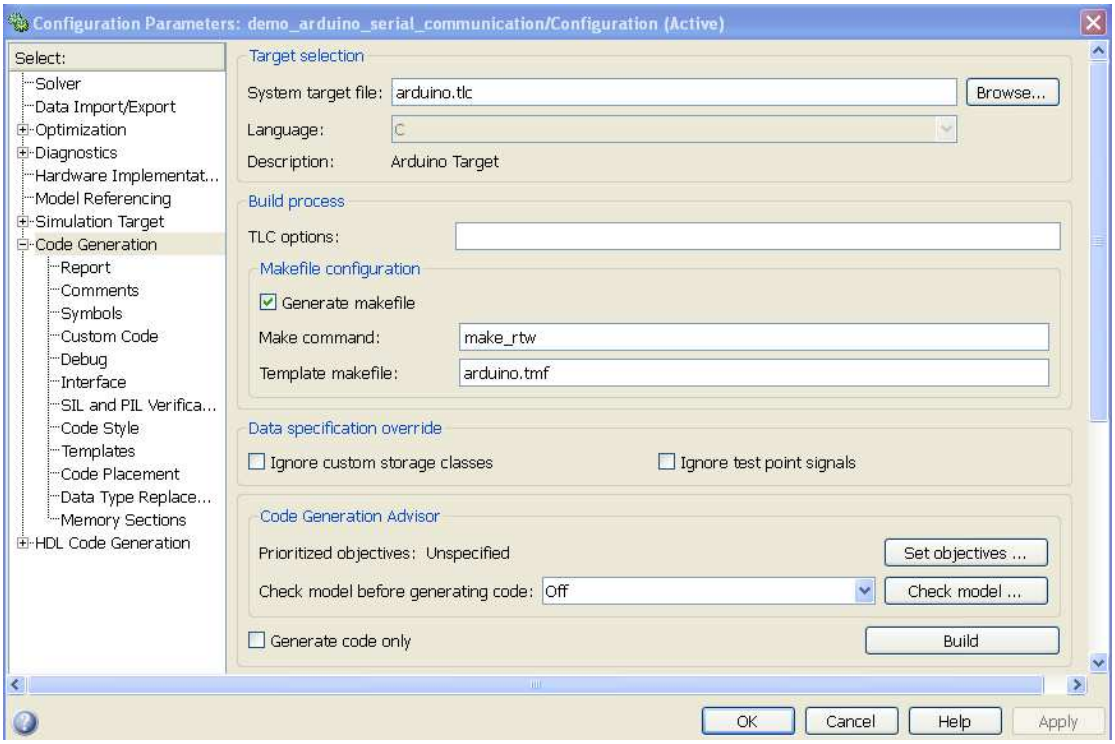
12.6 Для изучения, редактирования или компиляции кода прошиваемого в Arduino необходимо дважды щелкнуть по блоку “Lunch target side”.



Обратный переход в модель Simulink можно выполнить через активацию блока “Lunch host side” окна кода прошиваемого в Arduino.

Для успешной компиляции кода Arduino необходимо настроить следующие параметры.



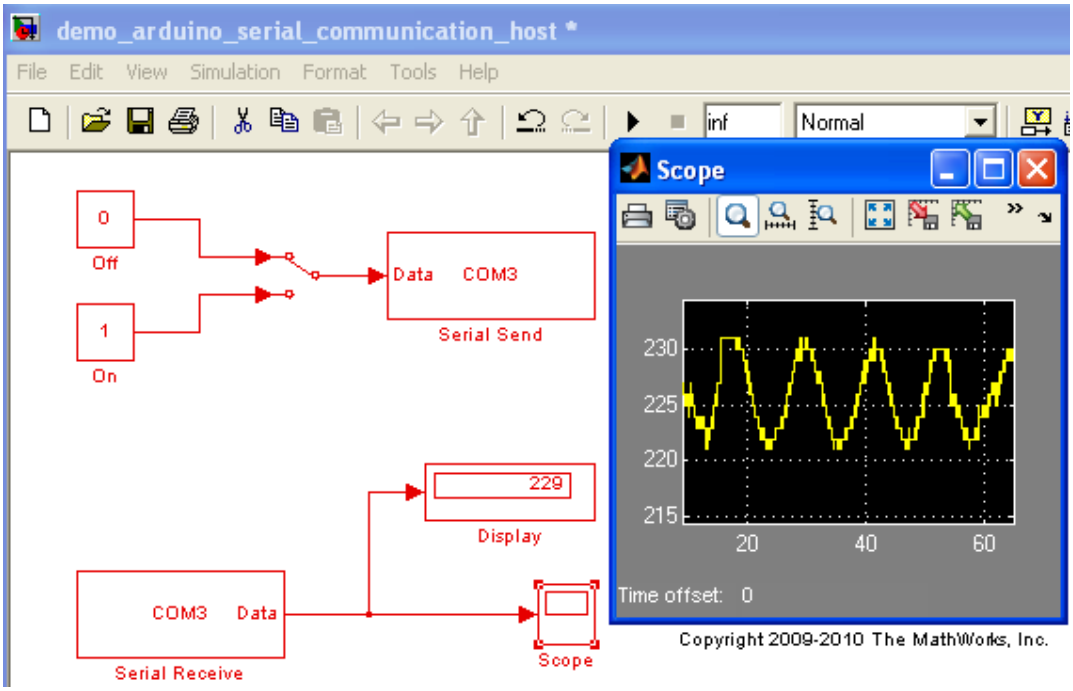


12.7 Комбинацией Ctrl+B или командой Tools>>Codegeneration>>BuildModel запустите кросс-компилятор.

Успешная компиляция сопровождается выводом следующего сообщения в командном окне.

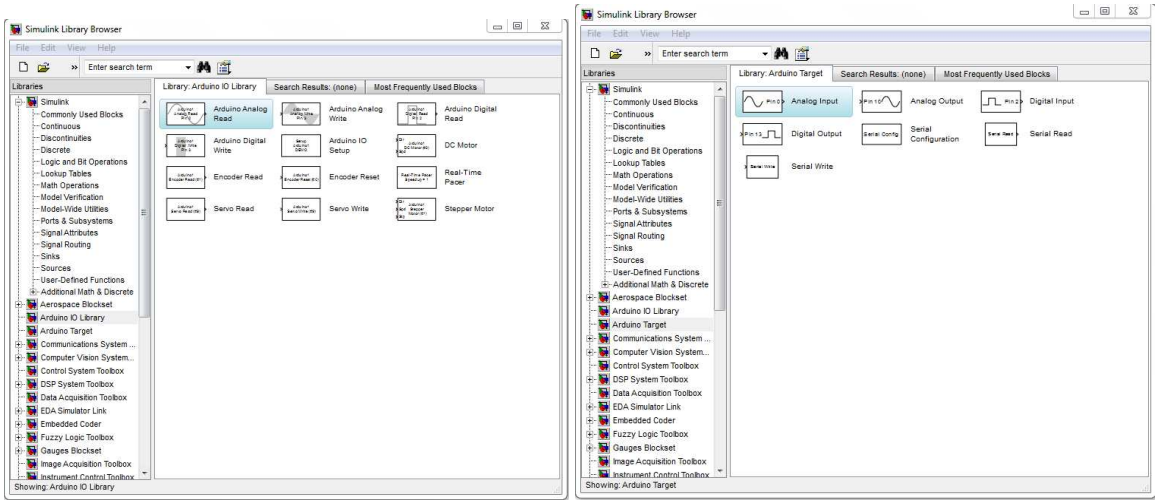
Successful completion of build procedure for model: demo_arduino_blink

12.8 Нажатием клавиши меню  загрузите код в Arduino и запустите Simulink модель на выполнение.



12.9 Поверьте возможность переключения светодиода Pin 13 arduino и отображение кода АЦП Pin 2 при помощи demo_arduino_serial_communication_host.mdl модели

13. Проверьте наличие разделов Arduino (Arduino IO Library и Arduino Target) в библиотеке Simulink:



Эти блоки могут быть использованы для написания кода прошиваемого в Arduino.

ПОДКЛЮЧЕНИЕ ARDUINO К LabVIEW

Для работы с платформой Arduino в LabView установите NI LabVIEW Interface for Arduino Toolkit http://sine.ni.com/nips/cds/view/p/lang/ru/nid/209835

Этот пакет имеет следующие особенности:

- Обеспечивает доступ к цифровым и аналоговым портам, ШИМ, интерфейсам I2C и SPI
- Обеспечивает работу с двигателями под управлением программ загруженных на Arduino
- Имеет примеры для решения разнообразных задач и ввода данных с датчиков
- Поддерживает работу с Arduino через USB, serial, Bluetooth или XBee
- Частота USB петли - 200 Hz, беспроводной - 25 Hz

ПРИМЕРЫ ПОЛУЧЕНИЯ ПРОВЕРЕННЫХ РЕЗУЛЬТАТОВ И ВАРИАНТЫ ДЛЯ САМОКОНТРОЛЯ

Задание 1. Программирование Arduino в интегрированной среде.

1. Запустите Arduino IDE –C:\arduino-1.0.3\arduino.exe
2. Настройте номер COM-порта среды (Tools > Serial port) на работу с COM портом Adruino из списка устройств компьютера
3. Выберите тип платы Tools > Board > Arduino UNO

4. Откройте  пример программы повторяющегося включения / выключения светодиода File > Open > c:\arduino-1.0.3\examples\01.Basics\Blink\Blink.ino и запустите  его.
5. По миганию светодиода убедитесь, что платформа Arduino работает.
6. Рассмотрите код программы.

ВОПРОСЫ:

- Каков размер прошитой программы?
- Каков размер памяти программ Arduino?
- Какова частота мигания светодиода?

7. Откройте и запустите пример File > Open > c:\arduino-1.0.3\examples\01.Basics\ AnalogReadSerial\ AnalogReadSerial.ino. Пример читает аналоговый вход N0 и выводит результат в канал последовательной передачи, связанный с COM портом компьютера.
8. Запустите программу COM Port Toolkit. Обратите внимание на поток принимаемых данных.
9. Сравните форматы следующих команд Arduino, передающие данные в канал COM порта компьютера: println; print, write.
10. Осторожно подсоедините аналоговый вход к земле.
11. Запишите значения считываемые из COM порта.
12. Не выключая COM Port Toolkit попробуйте снова выполнить программу Blink.ino
13. Закройте COM Port Toolkit и попробуйте снова выполнить программу Blink.ino

Задание 2. Связь платформы Arduino с МатЛАБ

1. Разработайте программу (m-код MatLAB) приема и отображения данных, принимаемых от Arduino через COM порт.
2. Разработайте программу (m-код MatLAB) переключения светодиода Arduino (pin 13).

Задание 3. Управление портом Arduino из Simulink

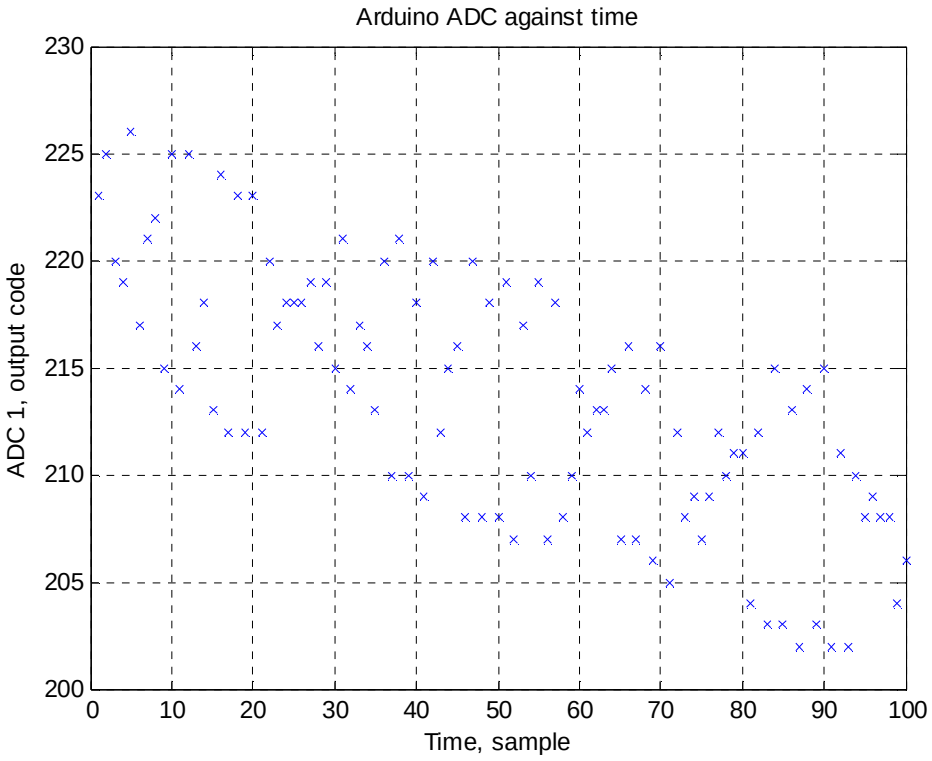
1. Разработайте Simulink модель приема и отображения данных, принимаемых от Arduino через COM порт.
2. Разработайте Simulink модель переключения светодиода Arduino (pin 13).

Задание 4. Ввод и отображение в МатЛАБ аналогового сигнала платформы Arduino

1. Подключите Arduino к MatLAB (R2008a или более поздней версии) как показано выше в разделе “Общие сведения”.

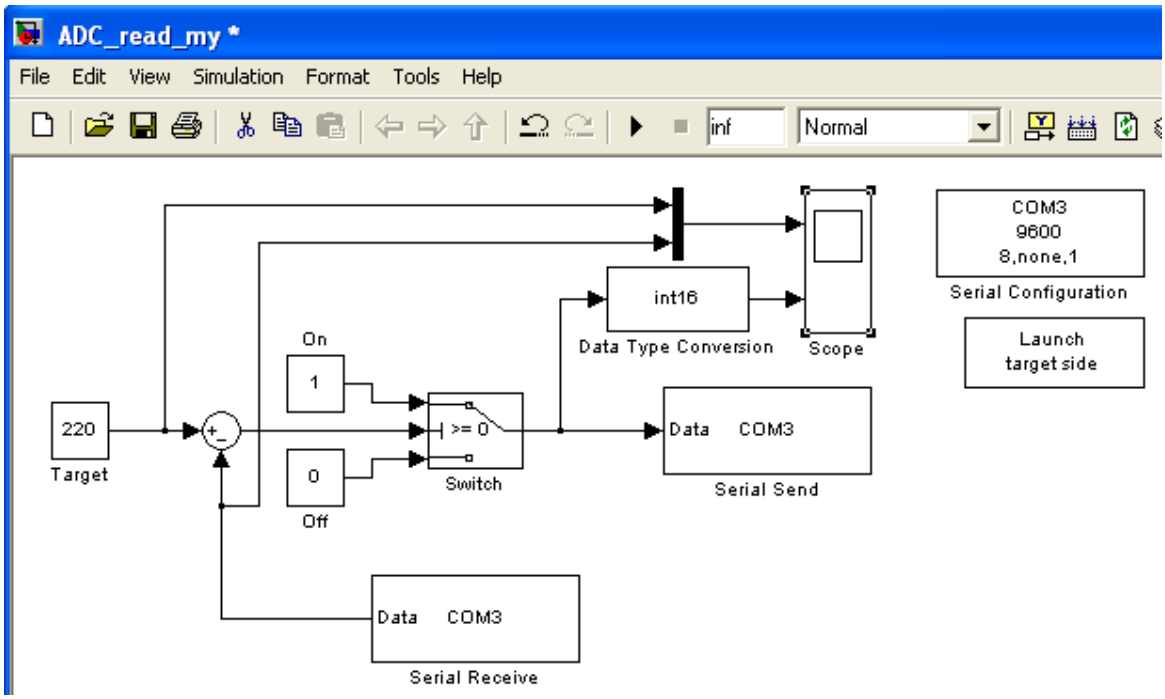
2. В среде MatLAB постройте осциллограф аналогового сигнала АЦП 1 платформы Adruino. Для решения задачи могут пригодиться следующие команды МатЛАБ: for, plot, pause, a.analogRead(1). Пример MatLAB кода и график результата показан ниже.

```
figure
for i = 1:100
    plot(i,a.analogRead(1),'xb');
    hold on;
end
grid
xlabel ('Time, sample');
ylabel ('ADC 1, output code');
title ('Arduino ADC against time')
```

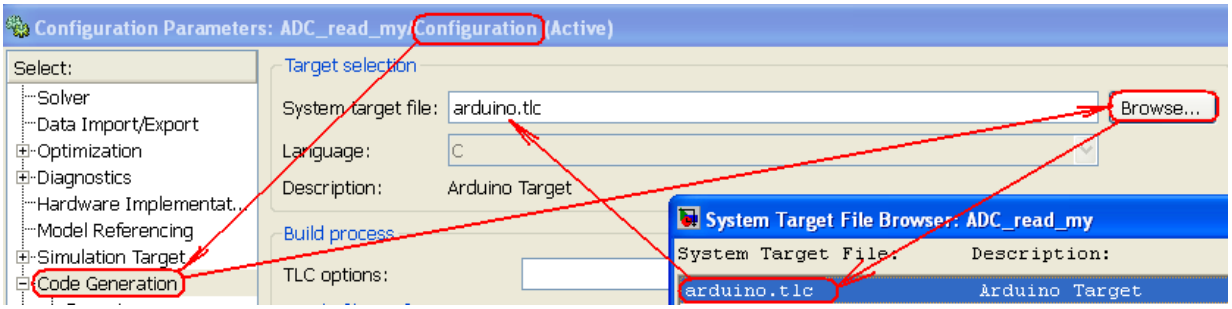


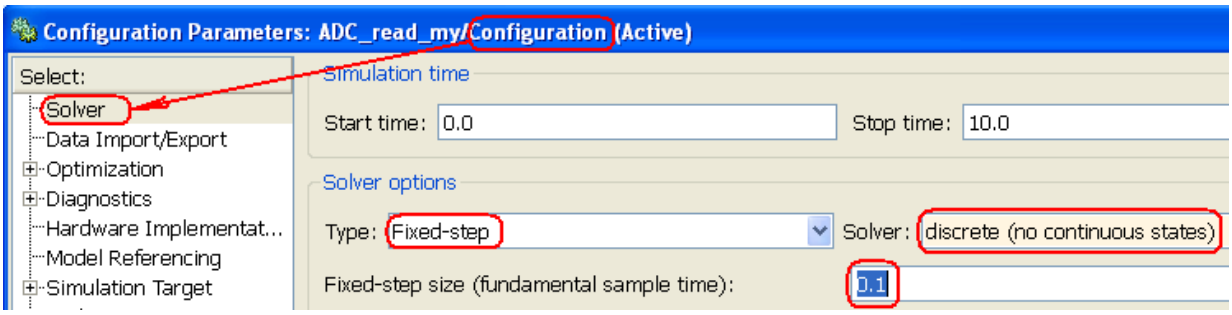
Задание 5. Создание интерактивной среды МатЛАБ – Simulink на базе платформы Arduino

1. Подключите Arduino к Simulink MatLAB (R2010a или более поздней версии) как показано выше в разделе “Общие сведения”.
2. Прошейте в Arduino модель simulink demo_arduino_serial_communication.mdl
3. Используя блоки библиотеки Simulink и модели demo_arduino_serial_communication_host.mdl соберите модель управления светодиодом с обратной связью, показанную ниже.

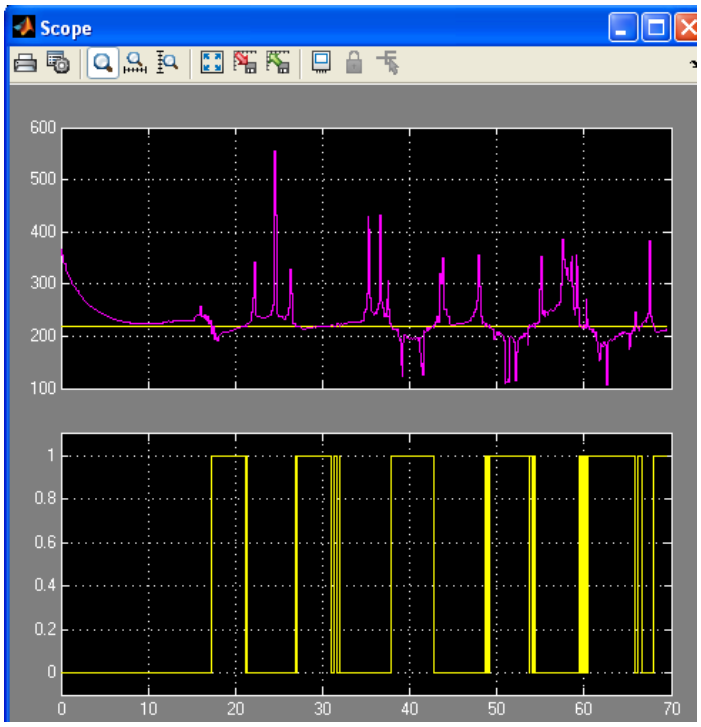


4. Настройте конфигурационные параметры вашей модели:





5. Запуская модель  и наблюдая за состоянием обратной связи, настройте значение блока Target так, чтобы светодиод включался когда уровень АЦП ниже заданного воздействия и выключался в противном случае, например, как показано ниже. Значение сигнала обратной связи можно изменять приближением / удалением пальца руки ко входу АЦП платформы Arduino.



6. Отсоедините USB кабель от Arduino.
7. Вместе с преподавателем подключите датчик температуры LM35 (5V – красный провод, выход 10 мВ/С - зеленый провод, земля – синий провод) к платформ Arduino.

LM35:

8. Доработайте последнюю модель:
9. Подключите Arduino USB кабелем к компьютеру.
10. Запустите модель.
11. Изменяя температуру на датчике убедитесь в работоспособности среды Simulink – платформа Arduino.

Задание 6. Построение системы реального времени Simulink – Arduino на базе таймера Arduino

1. Используя следующие примеры постройте систему, которая работает в режиме реального времени с тактом 0.1 сек. Система ручным переключателем модели Simulink управляет состоянием светодиода платформы (Pin 13), принимает код переключателя от Arduino в Simulink и отображает его.

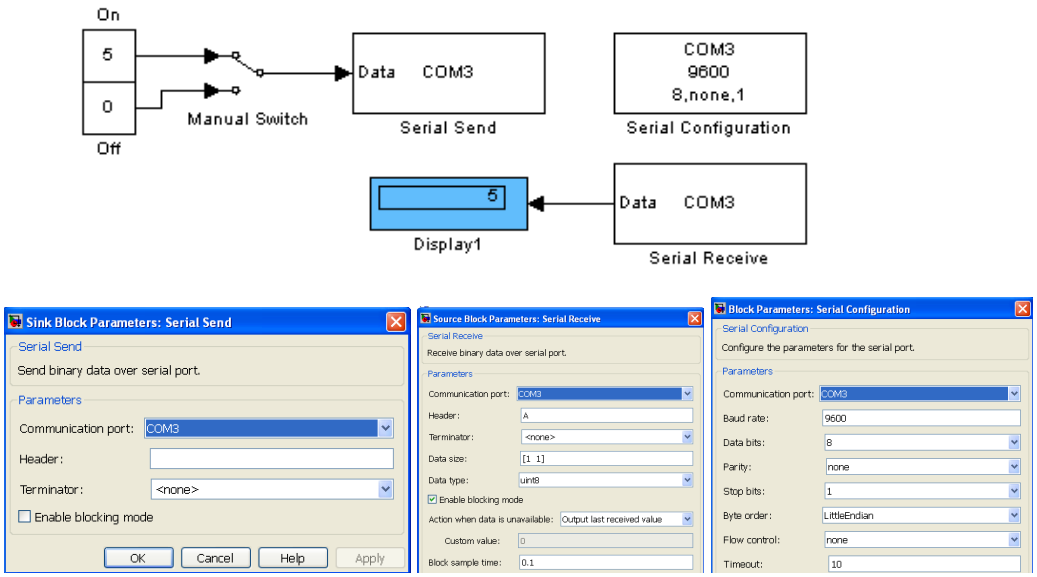
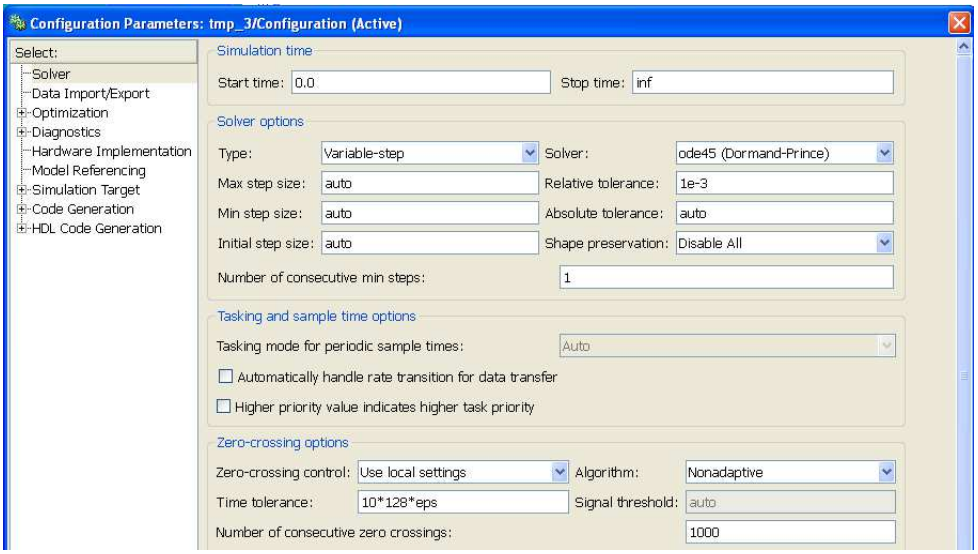
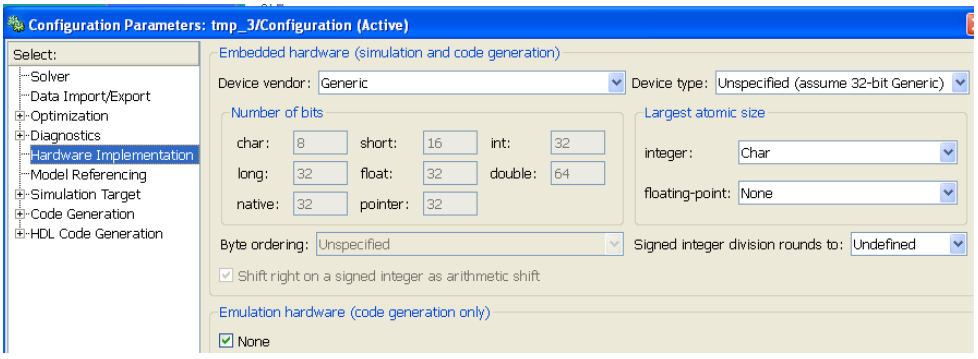


Рис. 3. Модель Simulink и параметры блоков модели.





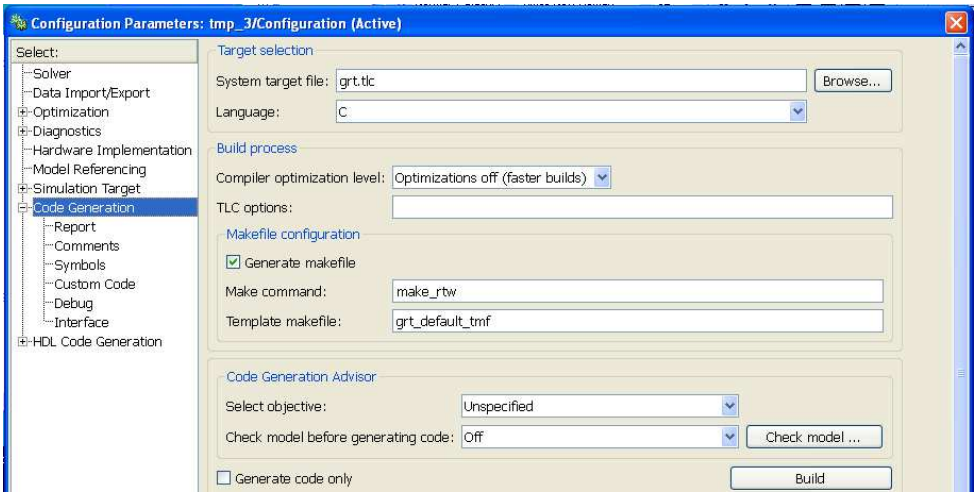


Рис. 4. Параметры конфигурации Модели Simulink.

Код RT программы Arduino.

```
/*
 Communication with Simulink model in RT mode

 created 20 April 2013
 modified
 by Bob Davidov

 This example code is in the public domain.
 */
int led = 13;
unsigned long set_time = 0;
int thisChar = 14;

void setup() {
  // Open serial communications and wait for port to open:
  Serial.begin(9600);
  Serial.flush();  // clear input buffer
  pinMode(led, OUTPUT);
}

void loop() {
  // get any incoming bytes:
  if (Serial.available() > 0) {
  thisChar = Serial.read();
  if (thisChar) digitalWrite(led, HIGH);
    else digitalWrite(led, LOW);
  }
  unsigned long time = millis();
  if (time > set_time) {
    set_time = set_time + 100;
    Serial.print("A"); // DEC, HEX, OCT, BIN, BYTE //8.599E+8
    Serial.write(thisChar);
```

```
//    Serial.write(12); // LOWBYTE
//    Serial.write(34); // HIGH BYTE
  }
}
```

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как обеспечить связь среды проектирования систем управления MatLAB и Smulink с внешней средой через платформу Ardulino ?
2. Как построить реальную систему термостатирования релейного типа на базе MatLAB, MatLAB-Smulink и платформы Ardulino ?
3. Какова скорость считывания АЦП платформы Arduino?
4. Какова максимальная частота контурного управления на базе платформы Arduino?
5. Почему при работающей программе COM Port Toolkit невозможно перепрограммировать Arduino?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. MathWorks. Arduino Support from MATLAB. http://www.mathworks.com/academia/arduino-software/arduino-matlab.html.
2. Arduino, site: http://www.arduino.cc/
3. Site RoboCraft: http://robocraft.ru/blog/projects/318.html
4. Site DFRobot: http://www.dfrobot.com/index.php