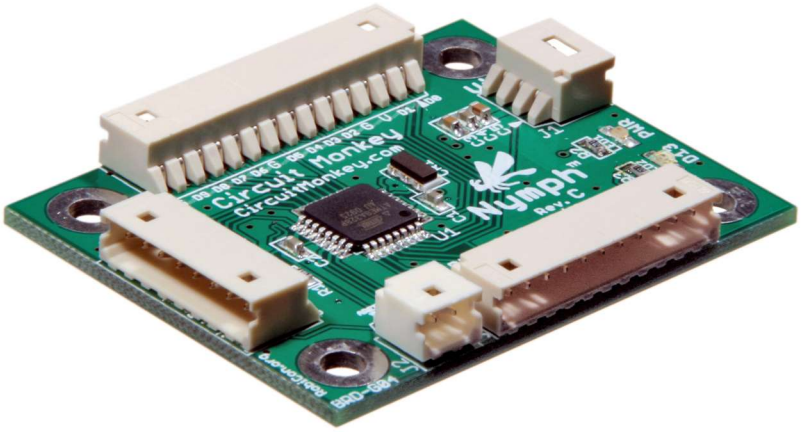
Circuit Monkey

500-0011-03

Nymph™

ATmega328P Based
Microcontroller
Rev. C

User Guide



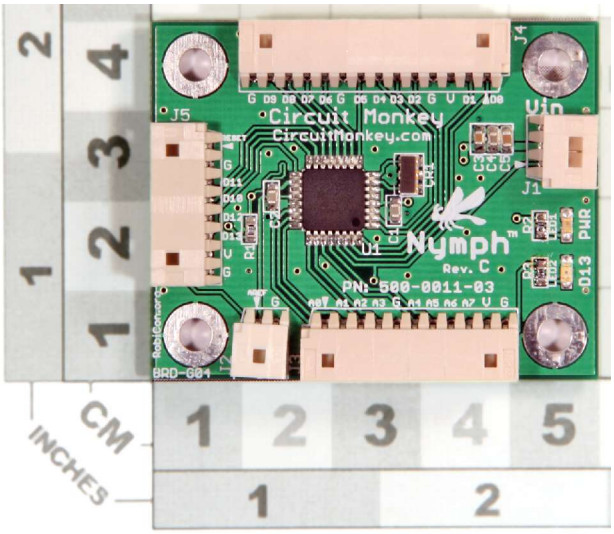
Version: 0.9
2009/09/02

Illustrations and photographs by Mark J Koch
This document was created with OpenOffice 3.0.

Table of Contents

Overview

Circuit Monkey [http://circuitmonkey.com] presents part number ***500-0011-03***, a microcontroller board based on the *Atmel Atmega328P* chip. The *Atmega328P* processor is popular among experimenters, hobbyists and engineers. The Input/Output signals on Nymph are connected using *Molex 53015 (MicroBlade)* series for reliable and vibration resistant connections. These connectors are *RobiCon.org* compliant *(an open-source interconnect standard that we are proposing)*. The board is 50mm x 40mm and features four mounting holes placed on a 40mm x 30mm mounting grid. The holes are plated and connected to logic ground and accept screws of size M3 or less *(M3, 6-32, M2.5, 4-40, etc.)*.

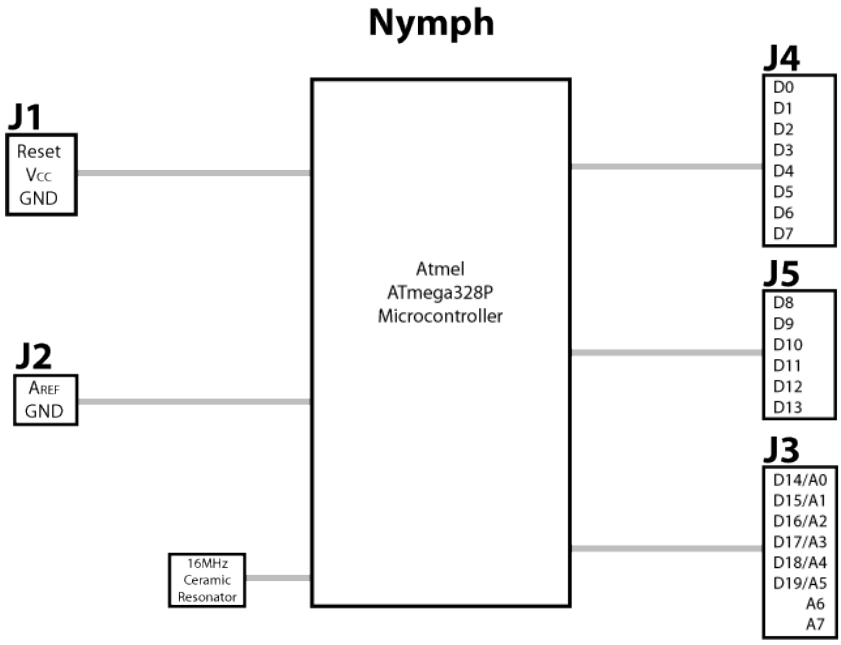


Features

- Power supply voltage; VCC = 5V (May be run as low as 1.8V under special conditions).
- CPU Operating Current: Icc=5mA(avg) / 9mA (max)
- Per I/O Pin Current: 40mA(max);
- Max Chip Current: Vcc-GND: 200mA (max)
- Power-save and power-down modes
- 16MHz on-board Ceramic Resonator.
- RobiCon.org compliant (Molex MicroBlade) connectors for all I/O and power connections.
- RobiCon.org compliant board outline and mounting.
- 50mm x 40mm x 8.1mm (WxDxH)
- 2.0"W x 1.6"D x 0.32"H
- 9.35g (0.33oz)

Block Diagram

A block diagram representing *Nymph* board is shown below.



Clock Selection

The *Nymph* controller comes with a 16MHz Ceramic Resonator on-board. Ceramic Resonators are electrically equivalent to Crystals but cost must less. However, resonators tend to be more sensitive to voltage and temperature and therefore are not generally used for timekeeping or precision timing applications as their frequency may vary slightly during use. The *Nymph* is intended as a hobby or pro-sumer device and therefore we feel that the resonator will give an amount of accuracy needed for most applications. A future revision of *Nymph* may allow for a Crystal based clock source. Please contact us if you have such a need in your design, product or project. The Nymph board comes pre-configured to use this clock source upon power up.

Alternately, the user may use the internal clock sources available inside the *Atmega328P* chip. Those sources are 1MHz, 4MHz and 8MHz. The use of these other timing sources may help with low power designs. These modes can be selected by programming the fuses inside the chip. Please consult the Atmega328P datasheet for details.

Reset Signal

The Reset signal is attached directly to the Reset input of the *Atmega328P* chip. We have added an on-board 10K ohm pull-up resistor. The user may actuate a reset state by simply shunting the Reset signal to *Ground* for a brief period of time. The reset signal is available on two connectors, J1 and J5, for external actuation.

SPI Connections

In normal use, J5 acts as an I/O connector allowing the user to use Port B[2..5] as needed. However, this connector can be used for SPI functionality. The Nymph board is programmed using J5. A typical SPI programmer connector uses the SCK, MISO, MOSI and Reset lines along with Vcc and Ground. As J5 contains all these signals, an adapter is available (or can be easily created) to attach to your favorite programmer device. Circuit Monkey offers an adapter suitable for the common 6-pin header style programmers. A comprehensive list of programmers can be found in Appendix A.

PWM and Timer Features

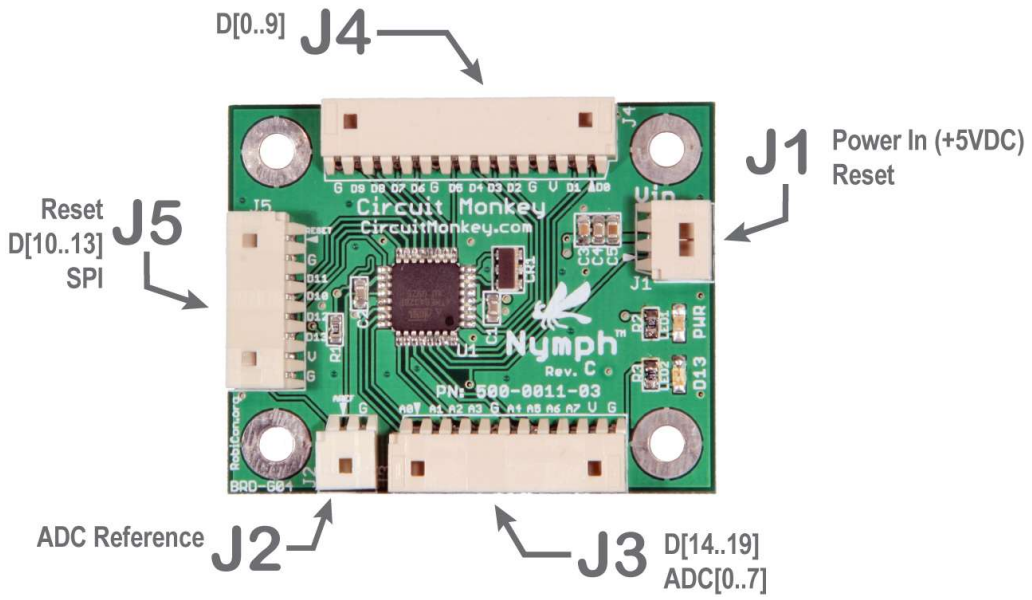
The Atmega328P chip comes with built in hardware timer circuitry for creating high frequency signals without the need for complicated software loops. For hobby and control applications most experimenters use this feature to control RC type servos or dimming LEDs. However, even through there are many IO pins on the Atmega328P, there are only 6 timers available and they have been designed by Atmel to be available on specific IO pins (D3, D5, D6, D9, D10, D11). The table below outlines the PWM pins. Notice that two pins (D.9 and D.10) are capable of 16-bit counter/timer operations. All other timers are 8-bit.

Data Line	ATmega 328P Function	Counter Size
D.3	OC2B	8-bit
D.5	OC0B	8-bit
D.6	OC0A	8-bit
D.9	OC1A	16-bit
D.10	OC1B	16-bit
D.11	OC2A	8-bit

Please consult the *Atmega328P* datasheet for full details. In addition we have found the following web site useful with getting started with PWM features on AVR chips:

http://www.8051projects.net/pulse-width-modulation/avr-pwm-example.php

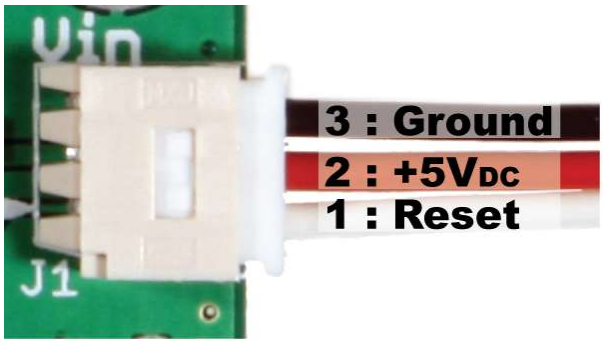
Connectors



J1 – Power and Reset

This connector features three contacts, Reset +5V and Ground. For most applications, provide power to the board by hooking +5VDC to pin 2 and Ground on Pin 3. Connection of Pin 1, Reset, is optional.

Pin	Description
1	Reset (active low)
2	Vcc (+5VDC)
3	Ground



J2 – Analog Converter Reference Voltage (optional)

This terminal block features two contacts, A_{REF} and *Ground*. This pin is connected directly to the Atmega328P AREF input. This pin is used to change the ADC reference voltage from it's built-in default. Please consult the Atmel Atmega328P datasheet for proper use of this signal.

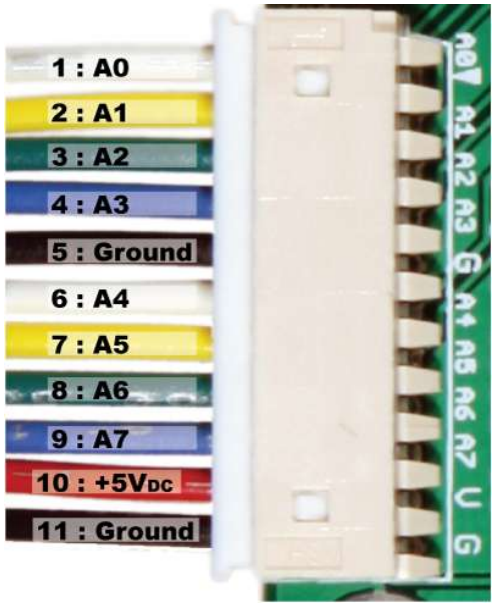
Pin	Description
1	AREF
2	**Ground**

J3 – Port_C[0..5] and/or Analog to Digital Inputs (ADC) [0..7]

This connector is a *Molex 53015-1110* and connects directly to Port C [0..5] and (ADC)Analog 6 and 7 on the Atmega328P chip. This connector also supplies one VCC (+5VDC) connection and two Ground connections. The pin-out is compliant with the RobiCon.org Type "M" 14-pin mixed-signal configuration.

J3 – Pin Out

Pin	Board Signal	ATmega 328P Signal	Description
1	*A0*	*PC0*	Port C.0 or ADC 0
2	*A1*	*PC1*	Port C.1 or ADC 1
3	*A2*	*PC2*	Port C.2 or ADC 2
4	*A3*	*PC3*	Port C.3 or ADC 3
5	***Ground***	***Ground***	Logic/Analog Ground
6	*A4*	*PC4*	Port C.4 or ADC 4
7	*A5*	*PC5*	Port C.5 or ADC 5
8	*A6*	*A6*	*ADC 6*
9	*A7*	*A7*	*ADC 7*
10	***Vcc***	***Vcc***	+5VDC logic supply
11	***Ground***	***Ground***	Logic/Analog Ground



J4 – Data[0..9] :: Port_D[0..7] and Port_B[0..1]

This connector is a *Molex 53015-1410* and allows access to all eight Port D I/O pins as well as two Port B [0 and 1] I/O pins. In our schematic, we refer to these lines as Data 0 through 9. This connector also supplies one VCC (+5VDC) connection and three Ground connections. The pin-out is compliant with the RobiCon.org Type "M" 14-pin mixed-signal configuration.

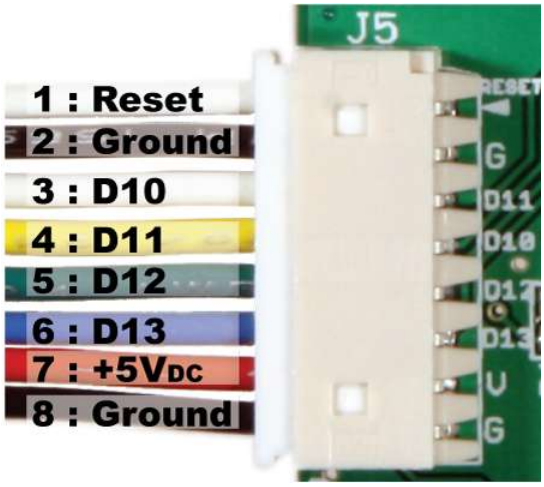
Pin	Board Signal	ATmega 328P Signal	Description
1	*D0*	*PD0*	Data 0 : Port D.0
2	*D1*	*PD1*	Data 1 : Port D.1
3	***Vcc***	***Vcc***	+5V DC logic supply
4	***Ground***	***Ground***	Logic Ground
1	*D2*	*PD2*	Data 2 : Port D.2
2	*D3*	*PD3*	Data 3 : Port D.3
1	*D4*	*PD4*	Data 4 : Port D.4
2	*D5*	*PD5*	Data 5 : Port D.5
4	***Ground***	***Ground***	Logic Ground
1	*D6*	*PD6*	Data 6 : Port D.6
2	*D7*	*PD7*	Data 7 : Port D.7
1	*D8*	*PB0*	Data 8 : Port B.0
2	*D9*	*PB1*	Data 9 : Port B.1
4	***Ground***	***Ground***	Logic Ground

J5 – Data[10..13] :: Port B[2..5] and SPI

This connector is a *Molex 53015-0810* and allows access to all eight Port B[2..5] I/O pins. This connector also acts as the SPI connection to the board. In our schematic, we refer to these lines as Data 10 through 13. This connector also supplies one VCC (+5VDC) connection and three Ground connections. The pin-out is compliant with the RobiCon.org Type "M" 8-pin mixed-signal configuration.

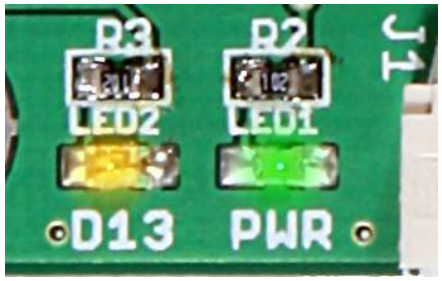
This connector is also used for programming operations using a SPI-based programmer.

Pin	Board Signal	ATmega 328P Signal	SPI Signal	Description
1	*RESET*	*RESET*	*RESET*	Reset (active low)
2	***Ground***	***Ground***	***Ground***	Logic Ground
3	*D10*	*PB2*	*SS*	Data 10 : Port B.2 : SS
4	*D11*	*PB3*	*MOSI*	Data 11 : Port B.3 : MOSI
5	*D12*	*PB4*	*MISO*	Data 12 : Port B.4 : MISO
6	*D13*	*PB5*	*SCK*	Data 13 : Port B.5 : SCK
7	***Vcc***	***Vcc***	***Vcc***	+5V DC logic supply
8	***Ground***	***Ground***	***Ground***	Logic Ground



Indicators

The Nymph board features two LEDs placed next to the J1 connector.



D13 LED

This LED is connected to Data line D13 (PB5) and can be programmed to turn on or off as needed. D13 is also the SCK line for SPI operations the LED will be on or flicker whenever the the board is being programmed or the user is using the SPI features of the board. For most applications it is useful to program it as a heart-beat or activity indicator .

Power-Good LED

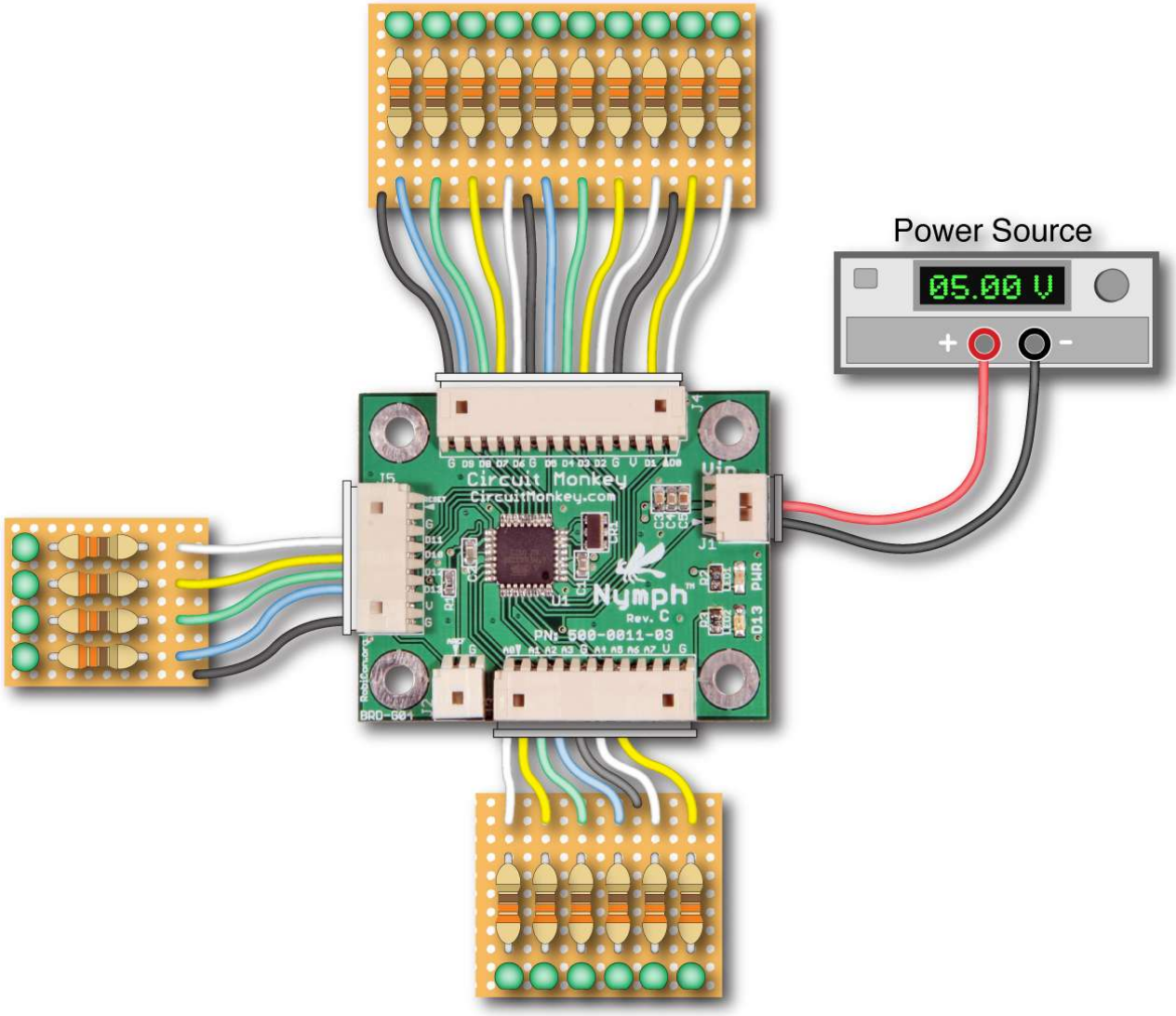
This LED is connected directly to the Vcc (+5VDC) supply. It will always light when you have power applied to the board.

Applications

LED Cycle Test

Hookup

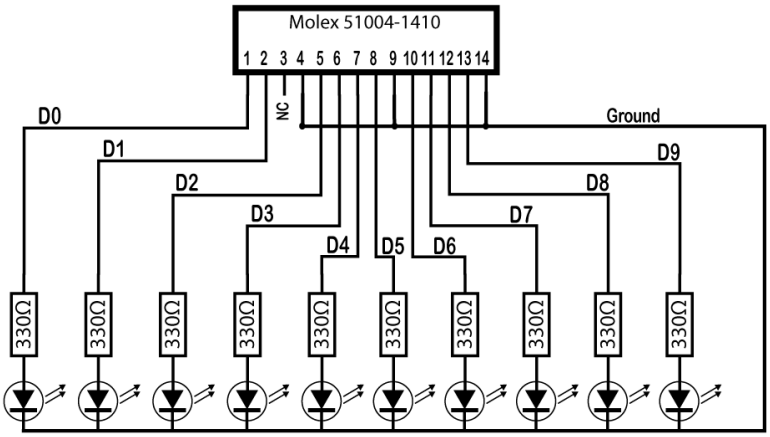
A simple test configuration diagram is shown below. Actual test code follows.



LED board diagrams

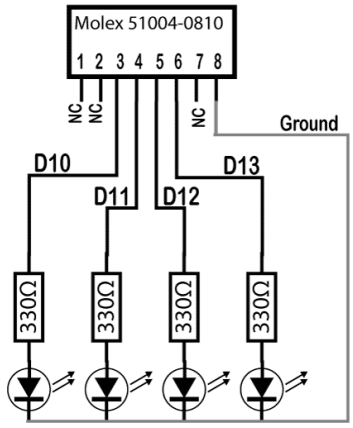
Below is a simple schematic for each of the LED test boards shown in the above hook-up diagram.

10 LED Board Schematic



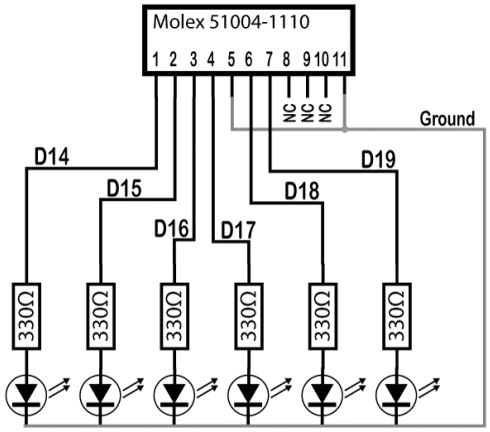
10 LED Test Circuit

4 LED Board Schematic



4 LED Test Circuit

6 LED Board Schematic



6 LED Test Circuit

LED Cycle Test Code (for Atmega328P – Nymph Compatible)

Using the preceding hookup diagram, one can test the functionality of the motor controller with the following code. It was designed to run on a *ATmega168/328* board. We tested with our own *Nymph* product but this code should also run on an *Arduino/Freeduino* or similar board, as they are also *Atmel AVR* based.

```
/**

  Circuit Monkey
  500-0011-03  --  Nymph Microcontroller Test/Demo

  Designed for Nymph (ATmega328) Microcontroller

  Author:  Mark J Koch

================================================================================
License:   BSD

Copyright (c) 2009, Circuit Monkey
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions  in binary form   must reproduce the  above copyright
      notice,  this list of conditions and  the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of Circuit Monkey  nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,  BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY   DIRECT,  INDIRECT,  INCIDENTAL,  SPECIAL,  EXEMPLARY,  OR
CONSEQUENTIAL  DAMAGES  (INCLUDING,  BUT  NOT  LIMITED TO,  PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES;  LOSS OF USE, DATA, OR PROFITS;  OR BUSINESS
INTERRUPTION)  HOWEVER CAUSED  AND ON ANY THEORY OF LIABILITY,  WHETHER IN
CONTRACT,  STRICT  LIABILITY, OR TORT  (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
================================================================================

 **/

#include <avr/io.h>
#define F_CPU 16000000UL  // 16MHz
#include <util/delay.h>

// Misc Defines
#define OFF     0
#define ON      1

//  Facade for _delay_ms
//  Helps reduce binary size since _delay_ms needs a float value
//  and we want to use an int value.  Calling the float version
//  all over your code would bloat your hex binary.
void delayMs(unsigned int ms)
```

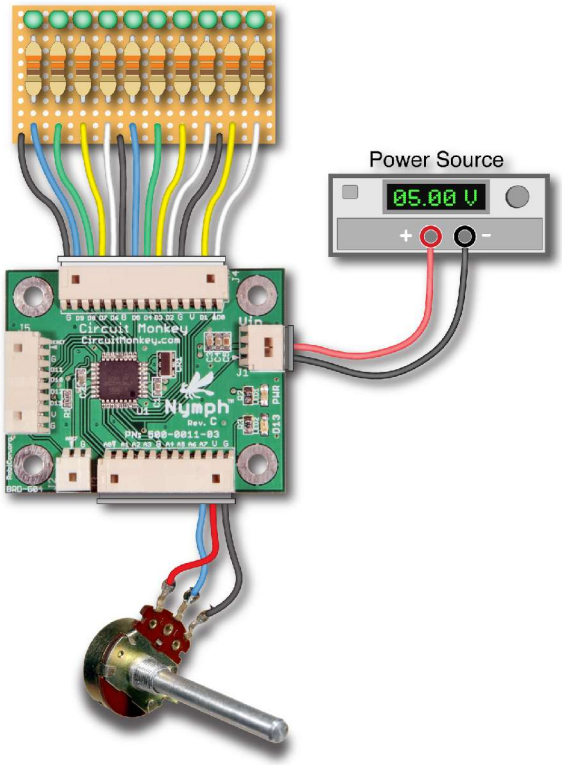
```
{
    while(ms){
        _delay_ms(0.96);
        ms--;
    }
}

void initPortB() {
    DDRB |= (1<<DDB0);
    DDRB |= (1<<DDB1);
    DDRB |= (1<<DDB2);
    DDRB |= (1<<DDB3);
    DDRB |= (1<<DDB4);
    DDRB |= (1<<DDB5);
}

void initPortC() {
    DDRC |= (1<<DDC0);
    DDRC |= (1<<DDC1);
    DDRC |= (1<<DDC2);
    DDRC |= (1<<DDC3);
    DDRC |= (1<<DDC4);
    DDRC |= (1<<DDC5);
}

void initPortD() {
    DDRD |= (1<<DDD0);
    DDRD |= (1<<DDD1);
    DDRD |= (1<<DDD2);
    DDRD |= (1<<DDD3);
    DDRD |= (1<<DDD4);
    DDRD |= (1<<DDD5);
    DDRD |= (1<<DDD6);
    DDRD |= (1<<DDD7);
}

void flashLED(int num) {
      switch(num) {
            case 0:
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
            case 6:
            case 7:
                  PORTD |= (1<<num);   // Set the bit
                  delayMs(100);
                  PORTD &= ~(1<<num);  // Clear the bit
                  delayMs(10);
                  break;
            case 8:
            case 9:
            case 10:
            case 11:
            case 12:
            case 13:
                  PORTB |= (1<<(num-8));      // Set the bit
                  delayMs(100);
                  PORTB &= ~(1<<(num-8));     // Clear the bit
                  delayMs(10);
                  break;
            case 14:
            case 15:
```

```
            case 16:
            case 17:
            case 18:
            case 19:
                  PORTC |= (1<<(num-14));      // Set the bit
                  delayMs(200);
                  PORTC &= ~(1<<(num-14));     // Clear the bit
                  delayMs(10);
                  break;
      }
}

int main(void)
{
    initPortB();
    initPortC();
    initPortD();

    int i;
    while (1) {
            for(i=0; i<20; i++) {
                  flashLED(i);
            }
    }
    // Never gets here.
    return(0);
}
```

Analog Input Potentiometer Test

Hookup

A simple test configuration diagram is shown below. Actual test code follows.

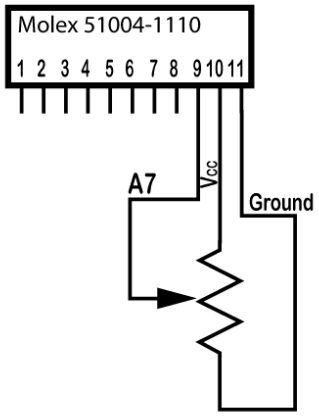


LED Board Diagram

See the previous section "LED Cycle Test" for the schematic of the 10-LED test board.

Potentiometer Diagram

The wiring for the analog input test device (a simple potentiometer) is shown below.



Analog Input Test Circuit

Analog Input Test Code (for Atmega328P – Nymph Compatible)

Using the preceding hookup diagram, one can test the functionality of the analog input with the following code. It was designed to run on a *ATmega168/328* board. This code should also run on an *Arduino/Freeduino* or similar board, as they are also *Atmel* ATmega based.

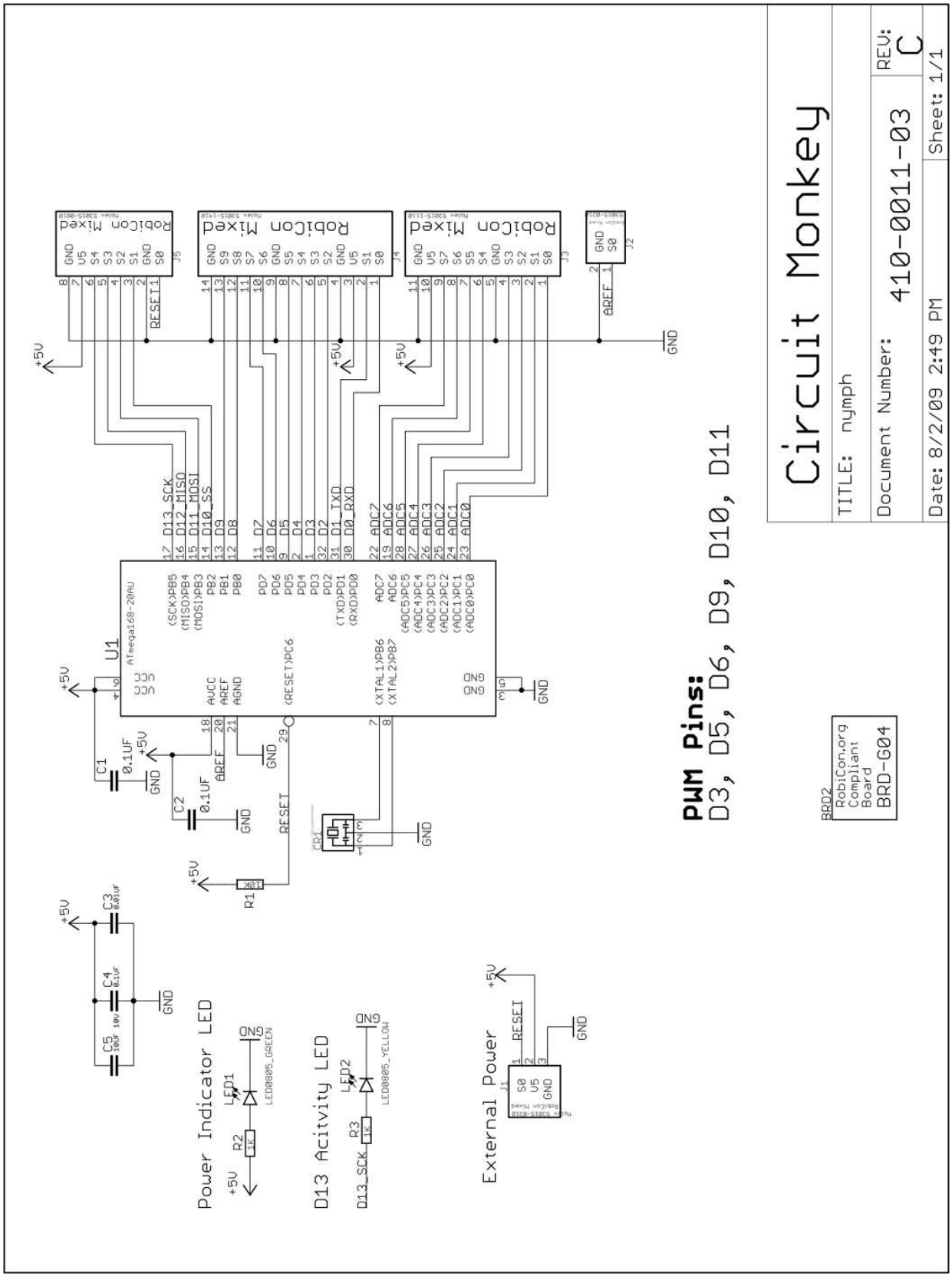
When properly hooked up and running, this code will light one of the 10 LEDs to indicate the position of the potentiometer. Note that we were tempted to light all the LEDs at full-scale but realized this might cause the Atmega328P to draw too much current (200mA max for chip).

```
/**
 **
  Circuit Monkey
  500-0011-03  --  Nymph Microcontroller Single Analog Input Test

  Designed for Nymph (ATmega328) Microcontroller

  Author:  Mark J Koch

================================================================================
License:   BSD

Copyright (c) 2009, Circuit Monkey
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

    * Redistributions of source code must retain the above copyright notice,
      this list of conditions and the following disclaimer.
    * Redistributions  in binary form   must reproduce the  above copyright
```

```
      notice,  this list of conditions and  the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * Neither the name of Circuit Monkey  nor the names of its contributors
      may be used to endorse or promote products derived from this software
      without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING,  BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
LIABLE FOR ANY   DIRECT,  INDIRECT,  INCIDENTAL,  SPECIAL,  EXEMPLARY,  OR
CONSEQUENTIAL  DAMAGES  (INCLUDING,  BUT  NOT  LIMITED TO,  PROCUREMENT OF
SUBSTITUTE GOODS OR SERVICES;  LOSS OF USE, DATA, OR PROFITS;  OR BUSINESS
INTERRUPTION)  HOWEVER CAUSED  AND ON ANY THEORY OF LIABILITY,  WHETHER IN
CONTRACT,  STRICT  LIABILITY, OR TORT  (INCLUDING NEGLIGENCE OR OTHERWISE)
ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
POSSIBILITY OF SUCH DAMAGE.
================================================================================
 **
 **/
#include <avr/io.h>

#define F_CPU 16000000UL  // 16MHz

#include <util/delay.h>

// Misc Defines
#define ADC_VREF_TYPE 0x40

//  Facade for _delay_ms
//  Helps reduce binary size since _delay_ms needs a float value
//  and we want to use an int value.  Calling the float version
//  all over your code would bloat your hex binary.
void delayMs(unsigned int ms)
{
    while(ms){
        _delay_ms(0.96);
        ms--;
    }
}

void initADC(void) {
    ACSR=0x80;
    ADMUX=ADC_VREF_TYPE;
    ADCSRA=0x85;
}

unsigned int readADC(unsigned char channel) {
      ADMUX=channel|ADC_VREF_TYPE;
      ADCSRA|=0x40; // Start the AD conversion
      while ((ADCSRA & 0x10)==0); // Wait for the AD conversion to complete
      ADCSRA|=0x10; // Stop the conversion
      return ADCW;
}

void initLEDs() {
    DDRD |= (1<<DDD0); // D0
    DDRD |= (1<<DDD1); // D1
    DDRD |= (1<<DDD2); // D2
    DDRD |= (1<<DDD3); // D3
    DDRD |= (1<<DDD4); // D4
    DDRD |= (1<<DDD5); // D5
    DDRD |= (1<<DDD6); // D6
    DDRD |= (1<<DDD7); // D6
```

```
    DDRB |= (1<<DDB0); // D8
    DDRB |= (1<<DDB1); // D9

}

void setLEDs(int num) {
      // ALl off
      PORTD &= 0x00;
      PORTB &= ~0x03;

      switch(num) {
            case 0:
            case 1:
            case 2:
            case 3:
            case 4:
            case 5:
            case 6:
            case 7:
                  PORTD |= (1<<num);      // Set the bit
                  break;
            case 8:
            case 9:
                  PORTB |= (1<<(num-8));  // Set the bit
                  break;
            default:
                  PORTB |= (1<<(1));      // Max scale
                  break;
      }
}

int main(void) {
    int adc;

    initLEDs();
    initADC();

    while (1) {
        adc = readADC(7);
        setLEDs(adc/114); // 114 = 1024 / 9
        delayMs(100);
    }
    // Never gets here.
        return(0);
}
```

Schematic


Circuit Monkey
TITLE: nymph
Document Number: 410-0011-03
REV: C
Date: 8/2/09 2:49 PM
Sheet: 1/1
PWM Pins:
D3, D5, D6, D9, D10, D11
BRD2
RobiCon.org
Compliant
Board
BRD-G04
U1
ATmega168-20AU
Robicon Mixed
Power Indicator LED
D13 Acivity LED
External Power
LED0805_GREEN
LED0805_YELLOW
RESET
D13_SCK
D12_MISO
D11_MOSI
D10_SS
D1_TXD
D0_RXD
AREF
+5V
GND

Appendix A: References and Links

AVR Programmers

Adafruit (AdaFruit.com)

From the AdaFruit website:

> "*USBtinyISP is a simple open-source USB AVR programmer and SPI interface. It is low cost, easy to make, works great with avrdude, has* ***both*** *6 and 10 pin standard ISP cables, is AVRStudio-compatible and tested under Windows and MacOS X*
>
> *Using this programmer and avrdude you can program any in-circuit "serial" programmable chip that avrdude supports (which is nearly all of them). It does not do JTAG or High Voltage programming. You can re-program Arduino's (and 'minimal arduinos') using this programmer.*
>
> *The project is based off of the USBtiny code & design.*"
>
> ~$22USD Product Link

Great programmer at a great price! Works on all major operating systems *(Windows, MacOS, Linux and Solaris)*.

Atmel

AVR In-System-Programmer (ISP) STK500	Product Link	~$35USD
AVRISP mkII In-System Programmer	Product Link	~$35USD

Circuit Monkey

Circuit Monkey is working on a USBTinyISP (AdaFruit clone) device. Our device will measure only 20mmx30mm and feature our open-source 8-pin Molex SPI connector (as found on the Nymph). Will work on *Windows, Mac, Linux and Solaris*.

Pololu

Pololu USB AVR Programmer

Product Link

USB to industry standard 6-pin SPI connector. Features built-in simple oscilloscope function *(requires additional software)*. Currently Windows only (does not work on Mac, Linux or Solaris).

AVR Programming Environments

Atmel AVR Studio

AVR Studio is the *Atmel* supported development environment and features many features to make programming the ATmega simple and easy. However, AVR studio is a Windows-only tool.

AVR Studio Link

Gnu avr-gcc compiler and tools

In the open-source world, there are free tools for developing code on the ATmega chips. Based on the widely used 'gcc' compiler there is a variant for AVR called 'avr-gcc'. The major difference most beginning users will find between avr-gcc and the Atmel AVR Studio is that avr-gcc does not have a GUI

AVR-LibC Home Page Link

Setup

Mac	http://www.ladyada.net/learn/avr/setup-mac.html
PC (WinAVR)	http://winavr.sourceforge.net/
Linux	http://www.linuxjournal.com/article/7289 http://www.tuxgraphics.org/electronics/200901/avr-gcc-linux.shtml
Solaris	Circuit Monkey Wiki -- Solaris GCC Setup Guide

Links

Circuit Monkey

http://www.circuitmonkey.com

RobiCon.org

A proposed open standard for Robotics Interconnect. Currently lead by the owner of Circuit Monkey.

http://www.robicon.org

Atmel

Manufacturers of the *AVR/ATmega* microcontroller chips.

http://atmel.com/products/AVR/

Ricky's World of Microcontrollers and Microprocessors

Great 8051/AVR PWM tutorial and more!

http://www.8051projects.net/pulse-width-modulation/avr-pwm-example.php