LAMPIRAN

Coding Pada Arduino Uno R3

```
/*
  Teleduino328EthernetClientProxy.ino - Teleduino328EthernetClientProxy example
  Version 328-0.6.9
  Nathan Kennedy 2009 - 2014
  http://www.teleduino.org

  This sketch is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
*/

#include <EEPROM.h>
#include <Servo.h>
#include <Wire.h>
#include <Teleduino328.h>

#include <SPI.h>
#include <Ethernet.h>

// User configurable variables
byteuseDhcp = true;
byteuseDns = true;
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
IPAddressdeviceIp(192, 168, 1, 100); // Only if useDhcp is false
IPAddressgatewayIp(192, 168, 1, 1); // Only if useDhcp is false
IPAddressdnsIp(192, 168, 1, 1); // Only if useDhcp is false
IPAddresssubnet(255, 255, 255, 0); // Only if useDhcp is false
IPAddressserverIp(173, 230, 152, 173); // Only if useDns is false
charserverName[] = "us01.proxy.teleduino.org"; // Only if useDns is true
unsignedintserverPort = 5353; // Can be set to either 53 or 5353
bytestatusLedPin = 8;
```

```
// User configurable key, this is used to authenticate with the proxy server
// This is checked against the EEPROM on boot and written if necessary
// The proxy server retreives the key from the EEPROM
byte key[] = { 0xC6, 0xBD, 0x91, 0x3B,
               0x56, 0x57, 0xE9, 0x35,
               0x55, 0x41, 0xEE, 0xBE,
               0x8C, 0xC2, 0xA3, 0x90 };

// Other required variables
byte data[257];
bytedataLength;
bytehexStage;
unsigned long lastInstruction = 0;
unsigned long lastRefresh = 0;
byte stage = 0;

// Declare client object
EthernetClient Client;

void setup()
{
  // Load presets
Teleduino328.loadPresets();

  // Set status LED pin
Teleduino328.setStatusLedPin(statusLedPin);

Teleduino328.setStatusLed(1); // Initialisation
  // Check the EEPROM header and check to see if the key is correct
  // This is to ensure the key is not cleared from the EEPROM
if(EEPROM.read(0) != '#')
  {
EEPROM.write(0, '#');
  }
```

```
if(EEPROM.read(1) != 0)
  {
EEPROM.write(1, 0);
  }
if(EEPROM.read(2) != '#')
  {
EEPROM.write(2, '#');
  }
if(EEPROM.read(160) != '#')
  {
EEPROM.write(160, '#');
  }
for(byte i = 0; i < 16; i++)
  {
if(EEPROM.read(161 + i) != key[i])
    {
EEPROM.write(161 + i, key[i]);
    }
  }
if(EEPROM.read(177) != '#')
  {
EEPROM.write(177, '#');
  }

  // Start network and attempt to connect to proxy server
Teleduino328.setStatusLed(2); // Network configuration
if(useDhcp)
  {
if(!Ethernet.begin(mac))
    {
Teleduino328.setStatusLed(2, false, 10000);
Teleduino328.reset();
    }
  }
```

```
else
  {
Ethernet.begin(mac, deviceIp, dnsIp, gatewayIp, subnet);
  }
delay(1000);

Teleduino328.setStatusLed(3); // Connect to server
if((useDns&&  !Client.connect(serverName,  serverPort))  ||  (!useDns&&  !Client.connect(serverIp,
serverPort)))
  {
Teleduino328.setStatusLed(3, false, 10000);
Teleduino328.reset();
  }
lastInstruction = millis();
}

void loop()
{
if(Client.connected())
  {
    // What we need to do depends on which 'stage' we are at
switch(stage)
    {
case 0: // Wait for start byte
if(Client.available())
        {
char c = Client.read();
if(c == '?')
          {
stage++;
          }
        }
break;
case 1: // Reset variables
```

```
dataLength = 0;
hexStage = 0;
stage++;
break;
case 2: // Instruction byte
if(Client.available())
      {
char c = Client.read();
if(c == '?')
        {
stage = 1;
break;
        }
else if(c == '\r' || c == '\n' || c == '.')
        {
stage = 0;
break;
        }
if(!hexStage)
        {
data[0] = Teleduino328.hexDecode(c) * 16;
        }
else
        {
data[0] += Teleduino328.hexDecode(c);
        }
hexStage = !hexStage;
if(!hexStage)
        {
stage++;
        }
      }
break;
case 3: // Data length byte
```

```
if(Client.available())
        {
char c = Client.read();
if(c == '?')
          {
stage = 1;
break;
          }
else if(c == '\r' || c == '\n' || c == '.')
          {
stage = 0;
break;
          }
if(!hexStage)
          {
data[1] = Teleduino328.hexDecode(c) * 16;
          }
else
          {
data[1] += Teleduino328.hexDecode(c);
          }
hexStage = !hexStage;
if(!hexStage)
          {
stage++;
          }
        }
break;
case 4: // Data
if(Client.available())
        {
char c = Client.read();
if(c == '?')
          {
```

```
stage = 1;
break;
        }
else if(c == '\r' || c == '\n' || c == '.')
        {
if(dataLength == data[1])
         {
stage++;
break;
         }
else
         {
stage = 0;
break;
         }
        }
if(!hexStage)
        {
data[2 + dataLength] = Teleduino328.hexDecode(c) * 16;
        }
else
        {
data[2 + dataLength] += Teleduino328.hexDecode(c);
        }
hexStage = !hexStage;
if(!hexStage)
        {
dataLength++;
        }
       }
break;
case 5: // Execute instruction and return result
Teleduino328.instruction(data);
Client.write('!');
```

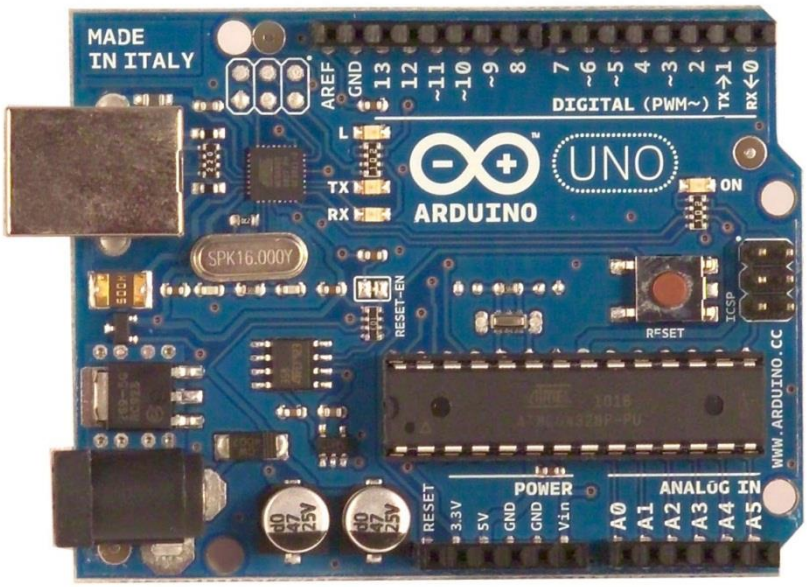
```
for(int i = 0; i < data[1] + 2; i++)
        {
Client.write(Teleduino328.hexEncode(data[i] / 16));
Client.write(Teleduino328.hexEncode(data[i] % 16));
        }
Client.write('\n');
lastInstruction = millis();
stage = 0;
break;
    }
  }
else
  {
Teleduino328.setStatusLed(10);
Teleduino328.reset();
  }

  // Has the instruction timeout been reached?
if(millis() - lastInstruction> 30000)
  {
Client.flush();
Client.stop();
Teleduino328.setStatusLed(9);
Teleduino328.reset();
  }

  // Process refreshes every 50ms
if(millis() - lastRefresh>= 50)
  {
Teleduino328.pinTimers();
Teleduino328.shiftRegisterTimers();
Teleduino328.shiftRegisters();
lastRefresh = millis();
  }
```

```
  // Check to see if reset has been requested
Teleduino328.checkReset();
}
```

Arduino UNO



Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serialdriverchip.Instead,itfeaturestheAtmega8U2programmedasaUSB-to-serialconverter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduinoboards,andthereferencemodelfortheArduinoplatform;foracomparisonwithpreviousversions, see the index of Arduinoboards.

Index

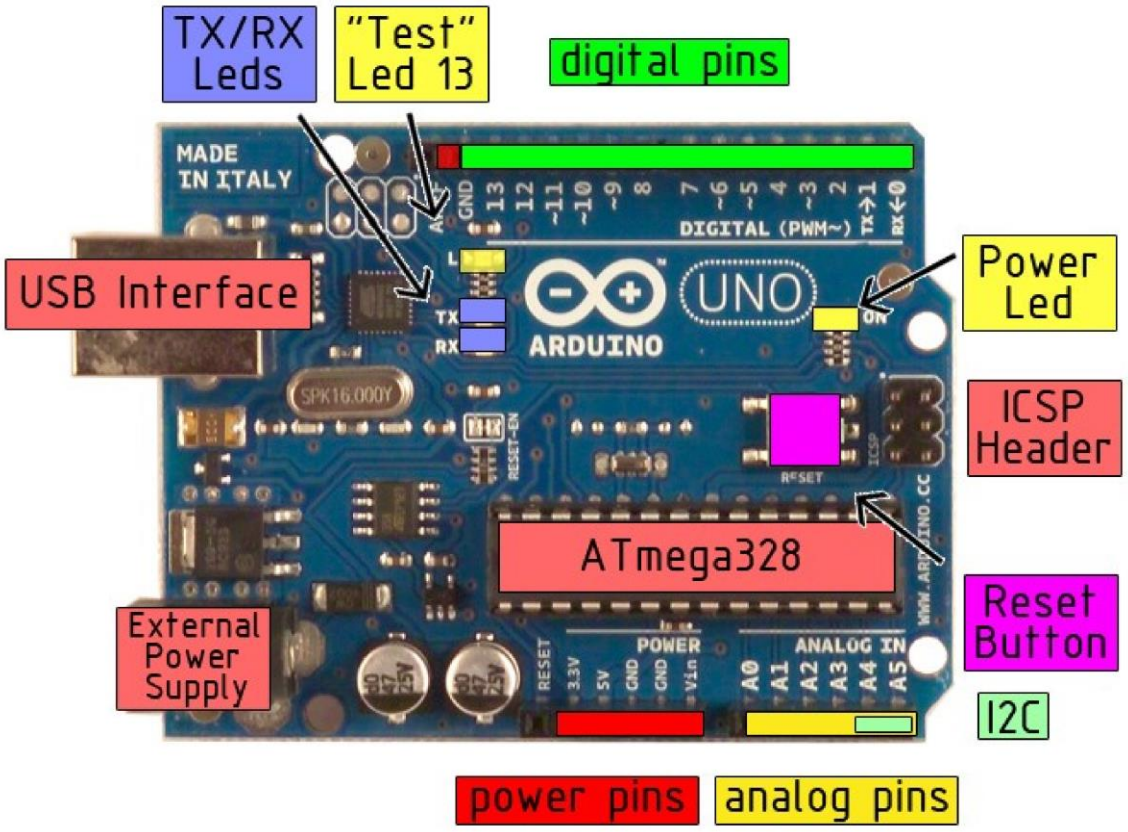
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
OperatingVoltage	5V
Input Voltage (recommended)	7-12V
InputVoltage(limits)	6-20V
DigitalI/OPins	14 (of which 6 provide PWMoutput)
AnalogInputPins	6
DC Current perI/OPin	40 mA
DC Current for3.3VPin	50 mA
FlashMemory	32 KB of which 0.5 KBusedby bootloader
SRAM	2KB
EEPROM	1KB
ClockSpeed	16MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through thispin,or,ifsupplyingvoltageviathepowerjack,accessitthroughthispin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5Vsupply.
- **3V3.**A3.3voltsupplygeneratedbytheon-boardregulator.Maximumcurrentdrawis50mA.
- **GND.** Groundpins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connectedtothecorrespondingpinsoftheATmega8U2USB-to-TTLSerialchip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a risingorfallingedge,orachangeinvalue.SeetheattachInterrupt()functionfordetails.
- **PWM:3,5,6,9,10,and11.** Provide8-bitPWMoutputwiththeanalogWrite()function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, althoughprovidedbytheunderlyinghardware,isnotcurrentlyincludedintheArduinolanguage.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it'soff.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I²C: 4 (SDA) and 5 (SCL).** Support I²C (TWI) communication using the Wirelibrary.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used withanalogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on theboard.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentationfor details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the referenceandtutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloaderthat allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFUbootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of theupload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending thisdata.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer ofprotection.IfmorethanmAisappliedtotheUSBport,thefusewillautomaticallybreaktheconnection until the short or overload isremoved.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language(based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook> Arduino-0017>Examples> Digital>Blink

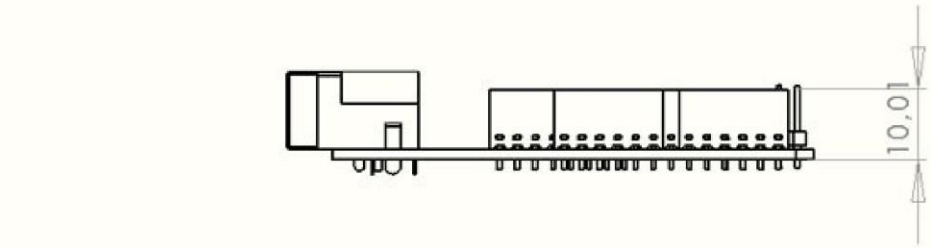
Once you have your skecth you'll see something very close to the screenshot on the right.

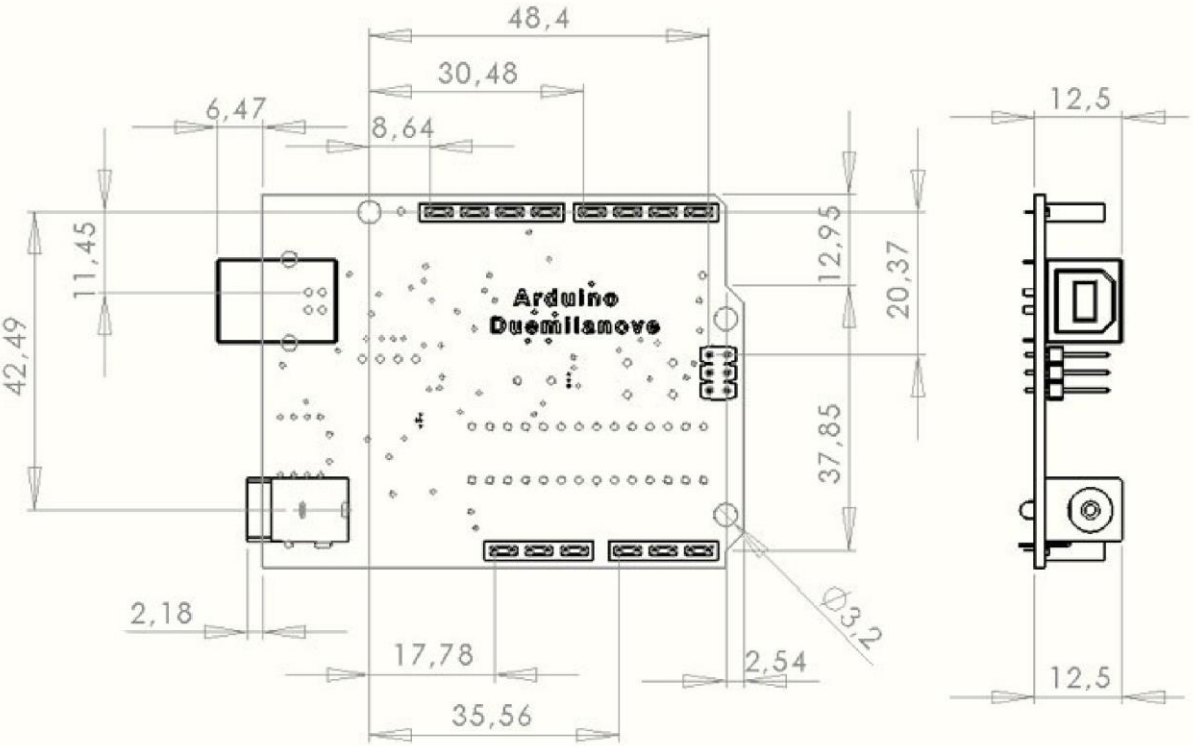
In **Tools>Board** select

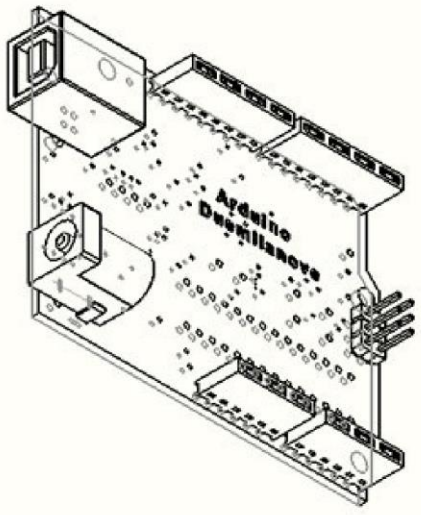
Now you have to go to

Tools>SerialPort
and select the right serial port, the one arduino is attached to.

Dimensioned Drawing







Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, orforanyproductsthathavebeenalteredormodifiedinanywaybyaCustomer.Moreover,Theproducershallnotbeliableforanydefectsthatresultfrom Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producershallhaveareasonabletimetoreplacements.Replacedproductsshallbewarrantedforanewfullwarrantyperiod.

1.3 EXCEPTASSETFORTHABOVE,PRODUCTSAREPROVIDED"ASIS"AND"WITHALLFAULTS."THEPRODUCERDISCLAIMS ALLOTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITYORFITNESSFORAPARTICULARPURPOSE

1.4 Customeragreesthatpriortousinganysystemsthatincludetheproducerproducts,Customerwilltestsuchsystemsandthefunctionalityofthe products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above,andnoadditionalobligationsorliabilitiesshallarisefromtheproducerprovidingsuchservices.

1.5 The Arduino™products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™products which is solely attheCustomer'srisk,andthatCustomerissolelyresponsibleforcompliancewithalllegalandregulatoryrequirementsinconnectionwithsuchuse.

1.6 Customeracknowledgesandagreesthatitissolelyresponsibleforcompliancewithalllegal,regulatoryandsafety-relatedrequirementsconcerningits products and any use of Arduino™products in Customer's applications, notwithstanding any applications-related information or support that may be provided by theproducer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilitiesandexpensesitincurstotheextentdirectlycausedby:(i)anactualbreachbyaCustomeroftherepresentationandwarrantiesmadeunderthis termsandconditionsor(ii)thegrossnegligenceorwillfulmisconductbytheCustomer.

3. Consequential DamagesWaiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

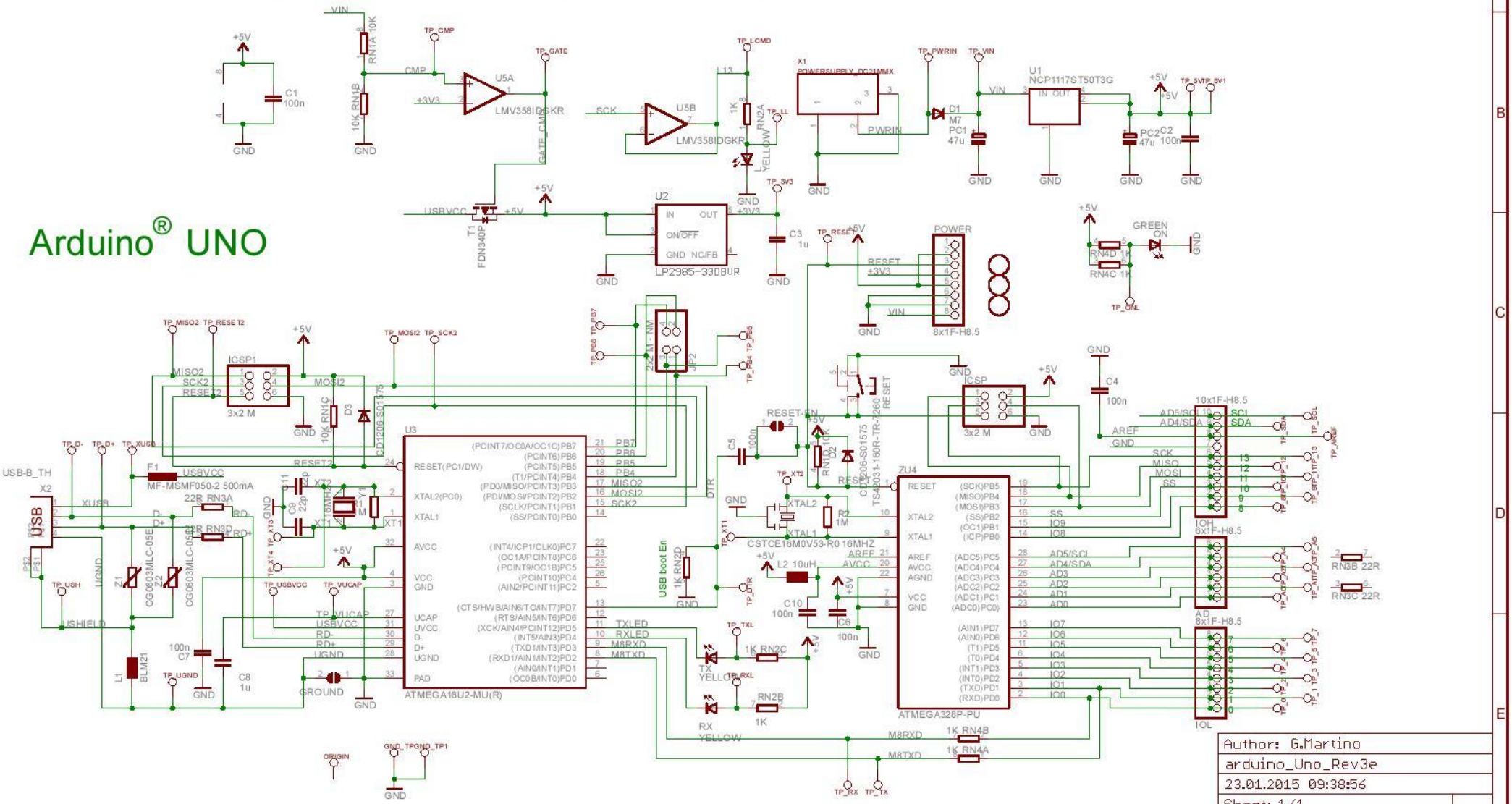
4. Changes tospecifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

Arduino® UNO



W5100 Ethernet Shield

-A high performance Ethernet shield for Arduino

Overview

W5100 Ethernet shield is a WIZnet W5100 breakout board with POE and Micro-SD designed for Arduino platform. 5V/3.3V compatible operation voltage level makes it compatible with Arduino boards, leafmaple, and other Arduino compatible board.

Features

- With Micro SDinterface
- 5V/3.3V double operational voltagelevel
- 10Mb/100Mb Ethernet socket withPOE
- All electronic brick interface are brokenout
- Operation temperature: -40℃ ~+85℃

Specifications

PCB size	55.88mm X 68.58mm X 1.6mm
Indicators	TX,RX,COL,FEX,SPD,LNK
Power supply	5V
Communication Protocol	SPI
RoHS	Yes

Electrical Characteristics

Specification	Min	Type	Max	Unit
Power Voltage	3V	-	5.5	VDC
Input Voltage VH:	3	-	5.5	V
Input Voltage VL:	-0.3	0	0.5	V
Current	-	-	100	mA

Hardware

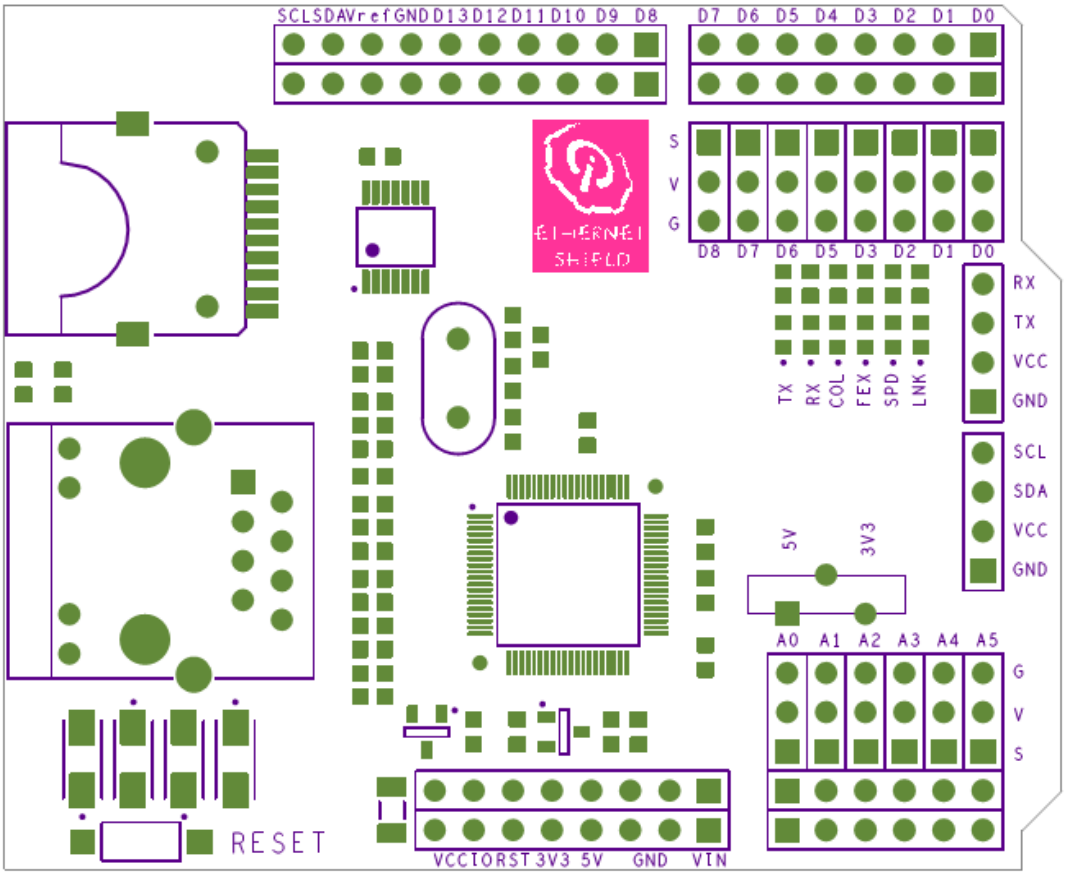


Figure 1 Top Map

Arduino PIN	Description
D0	Rx/Breakout
D1	TX/Breakout
D2	Breakout
D3	Breakout
D4	SD_CS
D5	Breakout
D6	Breakout
D7	Breakout
D8	Breakout
D9	W5100_Reset
D10	W5100_CS
D11	MOSI
D12	MISO
D13	SCK

A0	Breakout
A1	Breakout
A2	Breakout
A3	Breakout
A4	IIC_SDA/Breakout
A5	IIC_SCL/Breakout

Installation

When install W5100 Ethernet shield to Iteaduino, please check the operation voltage level of development board. If the voltage is 3.3V (IFLAT32,Leafmaple), set the Operation Level Setting switch to3.3V. IfthevoltageiS5V(Arduino),settheOperationLevelSettingswitchto 5V.

IteaduinocommunicateswithboththeW5100andSDcardusingthe SPI bus. This is on digital pins 11, 12, and 13 on the UNO/Duemilanove and pins 50, 51, and 52 on the Mega. On both boards,pin10isusedtoselecttheW5100andpin4fortheSDcard. ThesepinscannotbeusedforgeneralI/O.OntheMega,thehardware SSpin,53,isnotusedtoselecteithertheW5100ortheSDcard,but it must be kept as anoutput. NotethatbecausetheW5100andSDcardsharetheSPIbus,onlyone can be active at a time. If you are using both peripherals in your program, this should be taken care of by the correspondinglibraries. If you're not using one of the peripherals in your program, however, you'llneedtoexplicitlydeselectit.TodothiswiththeSDcard,setpin4asanoutputandwriteahightoit.Fort heW5100,setdigitalpin10 as a highoutput.

Inductor LED

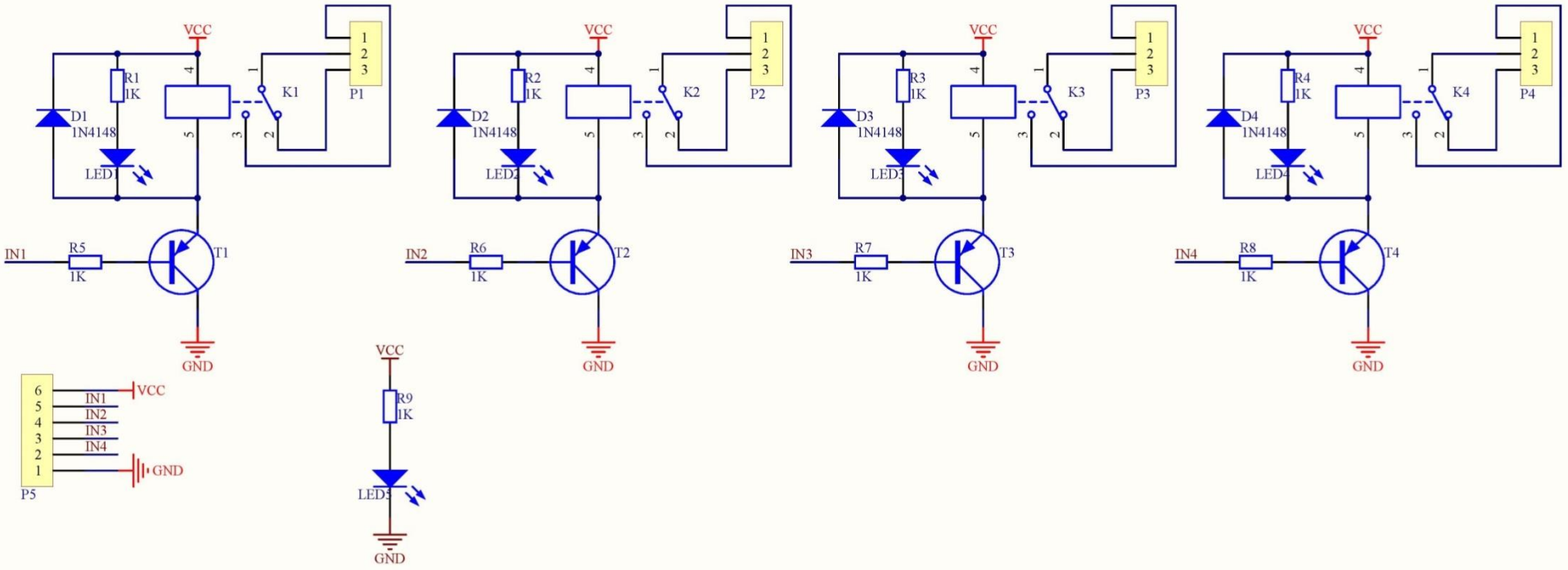
The shield contains a number of informational LEDs:

- LNK: indicates the presence of a network link and flashes when the shield transmits or receivesdata
- FEX: indicates that the network connection is fullduplex

- SPD: indicates the presence of a 100 Mb/s network connection (as opposed to 10Mb/s)
- RX: flashes when the shield receivesdata
- TX: flashes when the shield sendsdata
- COL: flashes when network collisions aredetected

RevisionHistory

Rev.	Description	Release date
v1.0	Initial version	2012-09-14



Title		
Size A4	Number	Revision
Date: 2012-2-28		Sheet of
File: D:\ \.\ .SchDoc		Drawn By: