Test interfaces for coreboot automated distributed test system

www.coreboot.org

v1.0

10 Oct 2013

Acknowledgements:

Thanks to Stefan, Dave, Marc, Peter and other folks at the #coreboot IRC channel for their continuous feedback. Special thanks to Google for promoting development of free and open source software through the Google Summer of Code (GSoC) program. This document is related to a project done for coreboot under GSoC 2013.

Feedback and questions:

IRC: http://www.coreboot.org/IRC

Mailing list: http://www.coreboot.org/Mailinglist

or visit www.coreboot.org to know more about coreboot project

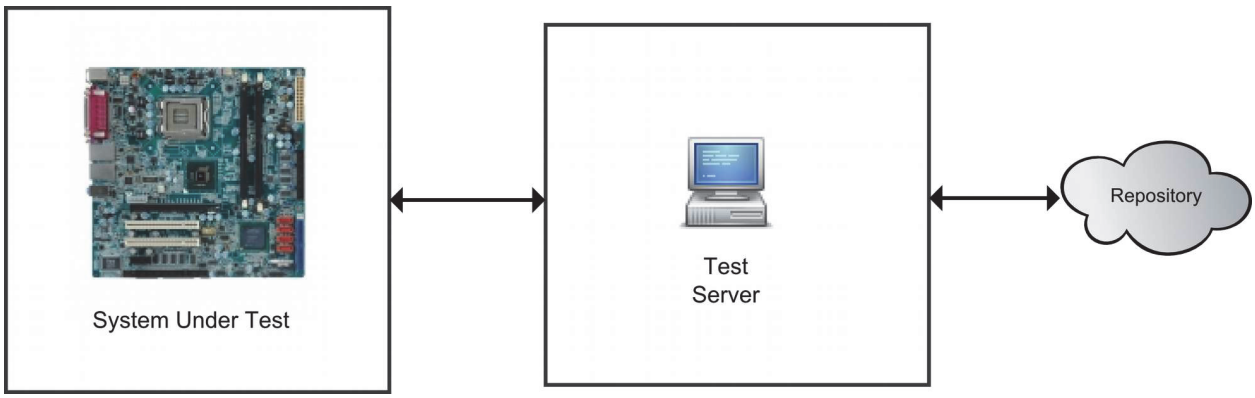
Contents

Change log

Date	Ver.	Author	Changes
10-Oct-13	1.0	Ayush Sagar	Initial

Motivation

Let's say we have a motherboard which is the system under test (SUT) and it's connected in 'some way' to a computer (test server) that performs tests on it. The computer has access to a local or remote coreboot development repository.



A simplistic view

To perform automated testing of a coreboot test build, it should follow a sequence like the following:

- Power off the SUT
- Get latest untested build from the repository and save it locally.
- Burn the coreboot test build on motherboard bios flash.
- Power on the SUT, perform basic tests, generate a standardized report file.
- While the SUT is on, dump it's serial port data (if available) on a file.
- Upload/save the test report and serial dump associated with the build in the same or separate repository.

To perform the above tasks there's a need to design the necessary software and hardware interfaces. Further it is desirable that these interfaces are efficient and extensible for different testing needs.

Developing the test interfaces

The test-server<->repository interface is mostly software based, but on the other hand the SUT<->test server interface is more demanding as it needs some hardware to be designed. Two devices have been specially designed to implement the SUT<->test server interface.

1. USB Power Strip: A controllable power strip to control mains power to the SUT.

The initial task is to power off the SUT completely by switching off the mains power. This is a required step to ensure that the SUT can be safely programmed. Powering off also helps to initialize the test conditions.

2. Test Interface Board: a way to talk to the SUT

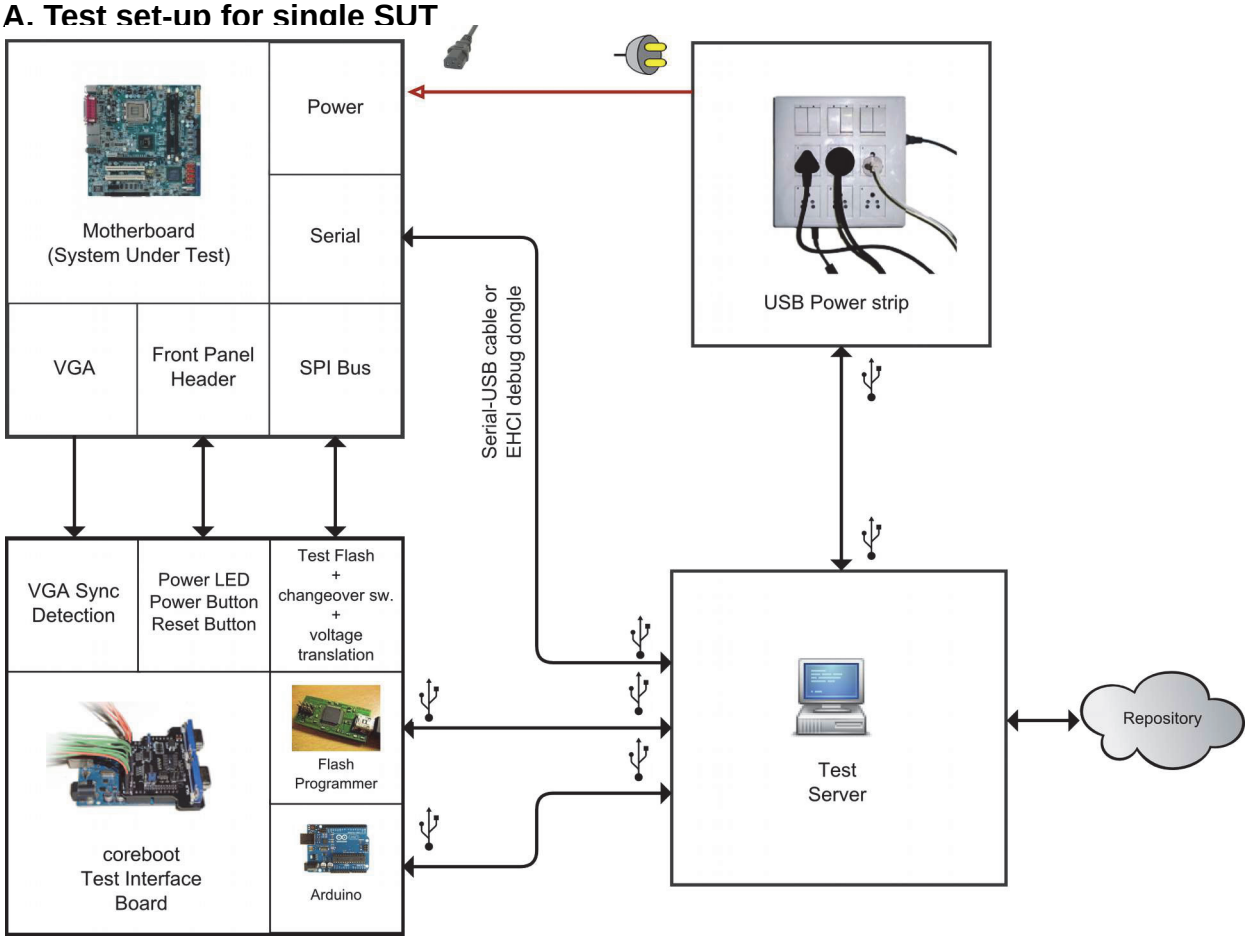
Many PC motherboards do not support In Circuit Programming (ICP) for their BIOS flash. Even in the supported boards the implementation of ICP does not follow a universal standard. The test interface board provides a standard ICP method for all kind of PC motherboards given that it uses SPI BIOS flash. It may be noted that only SPI serial flash is preferred since it has superseded other types. There are workarounds for parallel flash types but those are beyond the scope of this document.

In addition to ICP for SPI flash the test interface board also combines the following test functions:

- Power/Reset Button control
- Read Power LED pin to know the power state of the board
- Detect video signal presence to know if the motherboard is displaying anything when it's on
- Read the serial dump: from a serial port or by using the EHCI debug dongle

Possible test set-ups

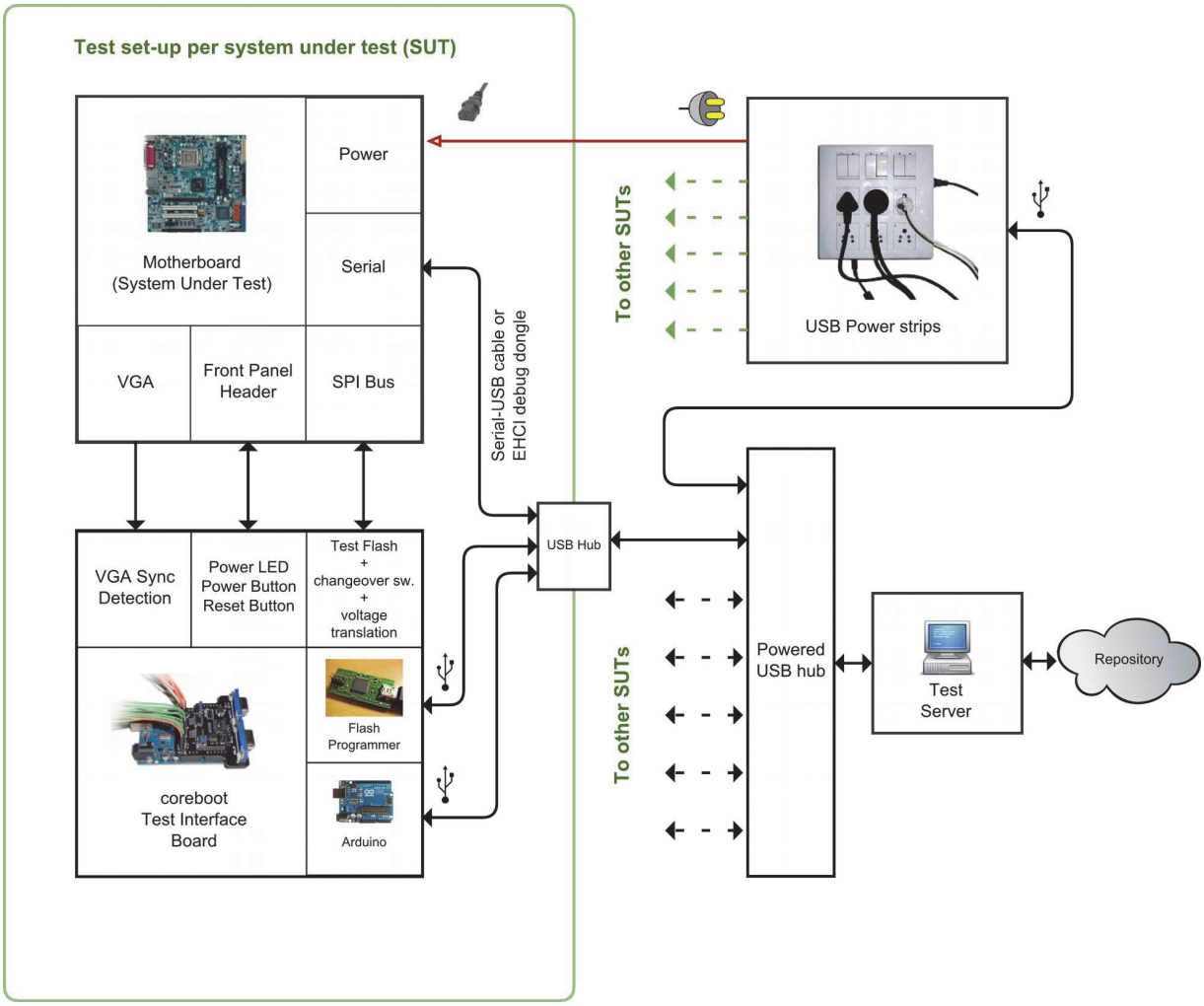
A. Test set-up for single SUT



A. Test set-up for single SUT

B. Test set-up distributed at test server level

We can extend this system to test multiple SUTs. In general such a system may look like the following:



B. Test set-up distributed at the test server level

Notice that this kind of set-up is distributed at test server level. It has a benefit of being less expensive when testing more SUTs at a local level. There are two challenges to the programmer in addition to the extra wires and USB hubs:

1. Requires ability to reliably identify each device uniquely

Each device is required to have a unique serial id string. The test interface board and the USB power strip are based around the Arduino Uno development boards which come with a unique serial id string from the factory. But there are two major road blocks:

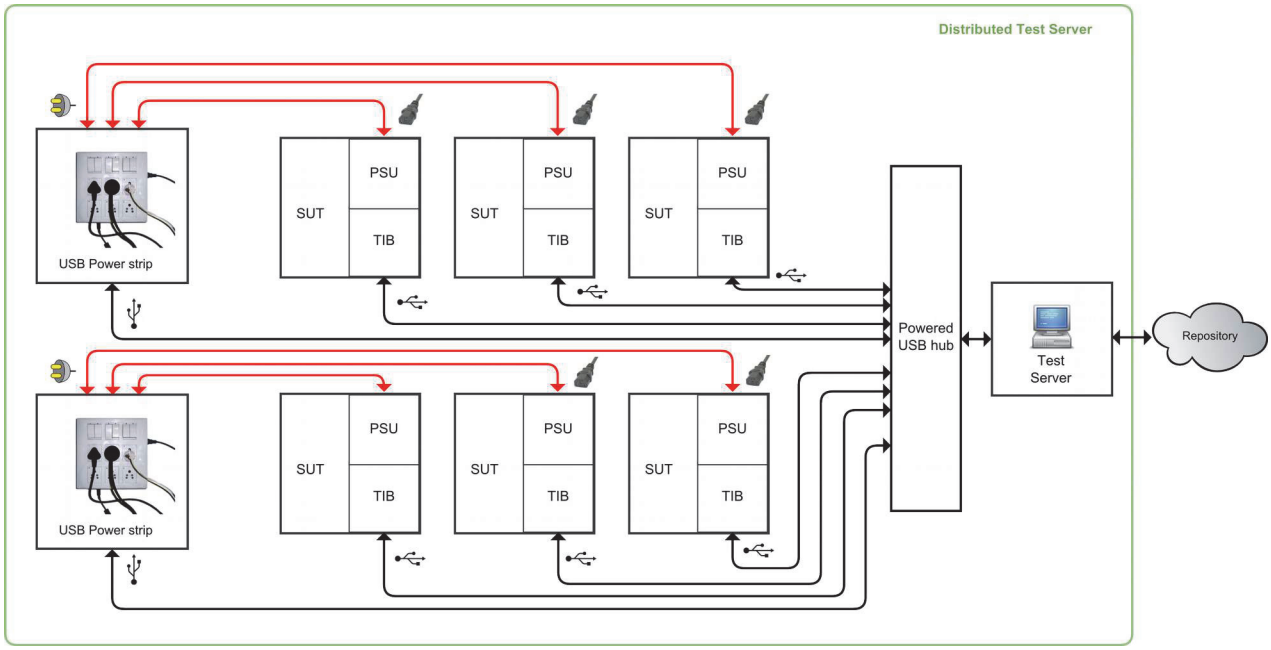
- The chosen flash programming devices need to have a unique serial id string otherwise this scheme won't work.
- A way to force flashrom (the open-source flashing software) to select from different devices hasn't been implemented yet. This needs to be done.

The test logic will be implemented in linux and most modern linux distributions use udev device manager. We can create udev rules to create bindings of these devices to symbolic names based on their USB device descriptors - VID, PID, and serial string.

2. Needs robust concurrency control

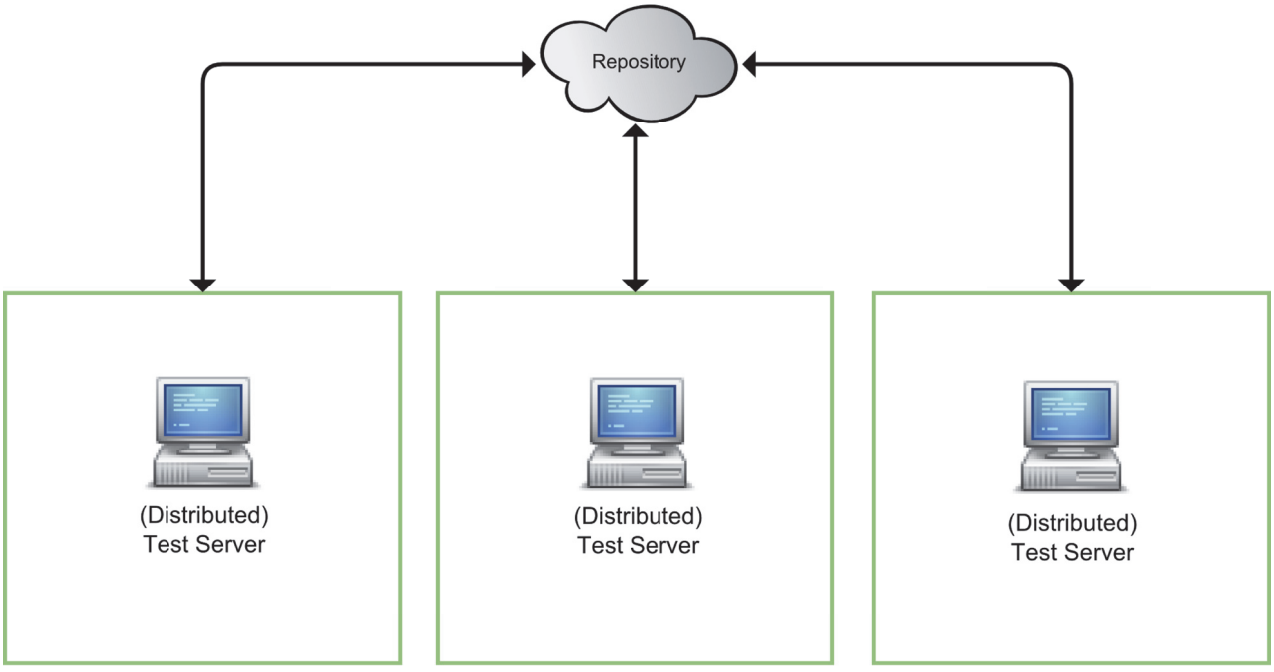
- The test logic must handle multiple operations well.
- Further, flashrom will need to be tested for reliability when running its multiple instances.

Here's an example of how this test set-up could look like for 6 SUTs.



An example of test set-up distributed at test server level containing 6 SUT

C. Test set-up distributed at repository level



C. Test set-up distributed at repository level

Following the similar trend we can extend our system in a way that multiple test severs can share a repository.

There can be two sub types:

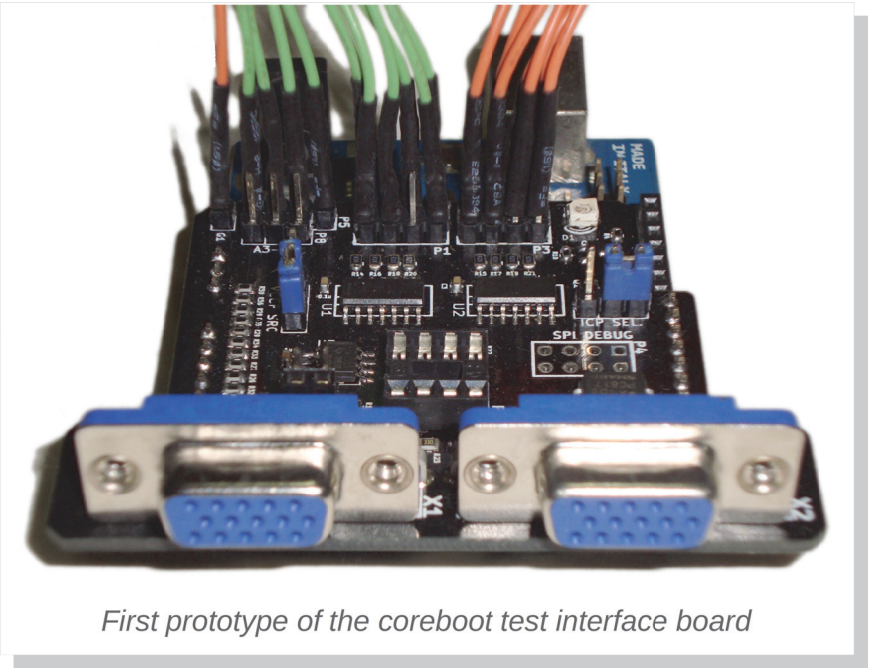
- **C1**: Test set-up with distributed test server and distributed repository.
- **C2**: Test set-up with non-distributed test-server and distributed repository.

The C2 type has a distinct advantage of making the system cost effective at small scale,

considering that the test routines are not very CPU intensive. We can use cheap single board computers like raspberry Pi. Moreover, a separate flash programming device is not required because the raspberry Pi has in-built flash programming capability. http://www.flashrom.org/RaspberryPi. It's readily implementable at present. The speed of flashing through raspberry Pi is slow however.

Coreboot Test Interface Board

Overview



First prototype of the coreboot test interface board

The coreboot test interface board (TIB) provides convenient electrical interfaces for programming/re-programming firmware on a PC motherboard and performing basic tests on it.

When combined with an Arduino Uno and suitable test logic on a PC, it can perform automated tests and automated SPI flash programming. The test logic is able to interact with this board using an included command line interface program.

Features

- Provides SPI In-Circuit-Programming (ICP) solution with support for logic level translation in the range 1.65V-5.5V to maximize compatibility among different motherboard and flash programming devices.
- Provides a stable and safer alternative to test-clip flashing.
- Augments test-clip flashing and provides a non-invasive method for testing coreboot on a motherboard using the HOLD function on the SPI flash. There's no need to remove the flash chip from the motherboard or to remove the stock firmware from it.
- Power and reset button control, power-on detection and video detection when combined with an Arduino Uno or a compatible board
- Test flash IC may be replaced conveniently on the test interface board. Test flash of different voltages can be used. *(Note: Flash V_{CC} must be equal to or lower than*

V_{CC} of motherboard SPI and programmer device used due to limitations of the translator IC)

- Protection against damage from electrostatic discharge (ESD) from normal handling on all ports.

The test interface board when coupled with an Arduino and connected to a computer provides the following services:

- Query/set ICP mode: if ICP jumper set to internal.
- Report whether VGA sync is present by measuring vertical and horizontal sync frequencies and checking whether it lies in the acceptable range. A frequency counter has been implemented using the hardware interrupts.
- Query power LED state to know whether the board has been successfully turned on/off.
- Operate Power and Reset buttons.
- Query VCCP, VCCF, VCCM to check for correct operation of the system. It uses statistical mode filter to filter out intermittent behavior of the ADC and the readings are referenced to the Atmega's internal voltage reference instead of the power supply. So it gives accurate voltage reading reliably even with a single query.
- Query A3, A4, A5 analog inputs. These are extra ADC input for measuring any voltage in range 0 – 24.5VDC for Arduino Uno/Duemilanove and 0 – 56.5 VDC for Arduino Leonardo

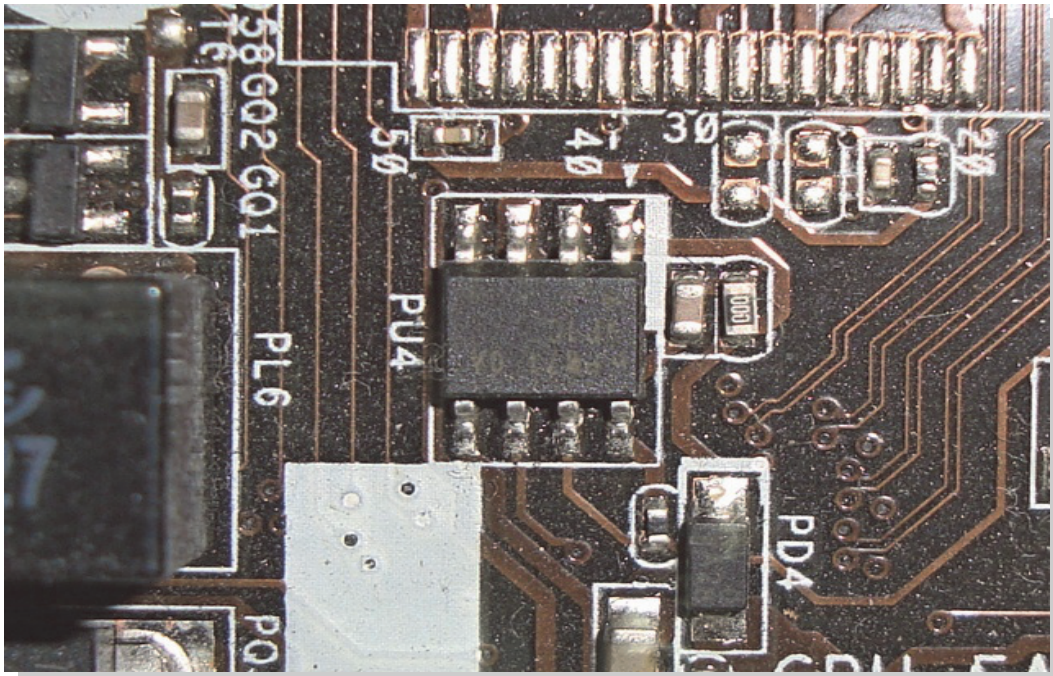
Usage

Connecting to motherboard's SPI bus (P1, P2, P3, JP1, JP2)

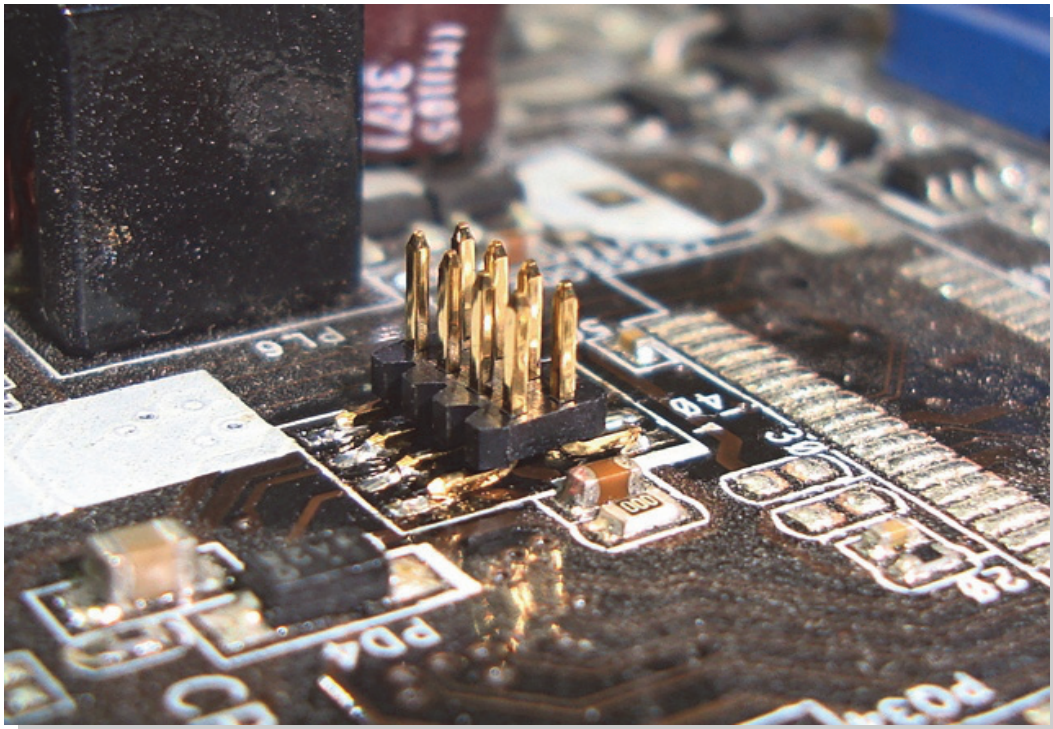
There are two ways to connect a motherboard's SPI bus to the test interface board. First is the *fixed method* that requires de-soldering the flash from the motherboard if it doesn't have a socket and soldering a male SMD header on its place to make an easy connection with the test interface board. Second is the test clip method which is non-invasive. It requires connecting a test-clip over the flash IC and enabling the SPI hold on the flash so that the motherboard talks only to the test flash. Here is a comparison:

	Fixed method	**Test clip method**
Ease	Requires de-soldering flash chip and then soldering a header on its footprint if it's a socket-less board	Quick and easy
Long term convenience	Convenient. Electrically and mechanically more stable	Less convenient in long term. Not stable.
Suitability	When performing repeated testing over a longer time on the same board	Good for performing quick tests and when a non-invasive method is required.

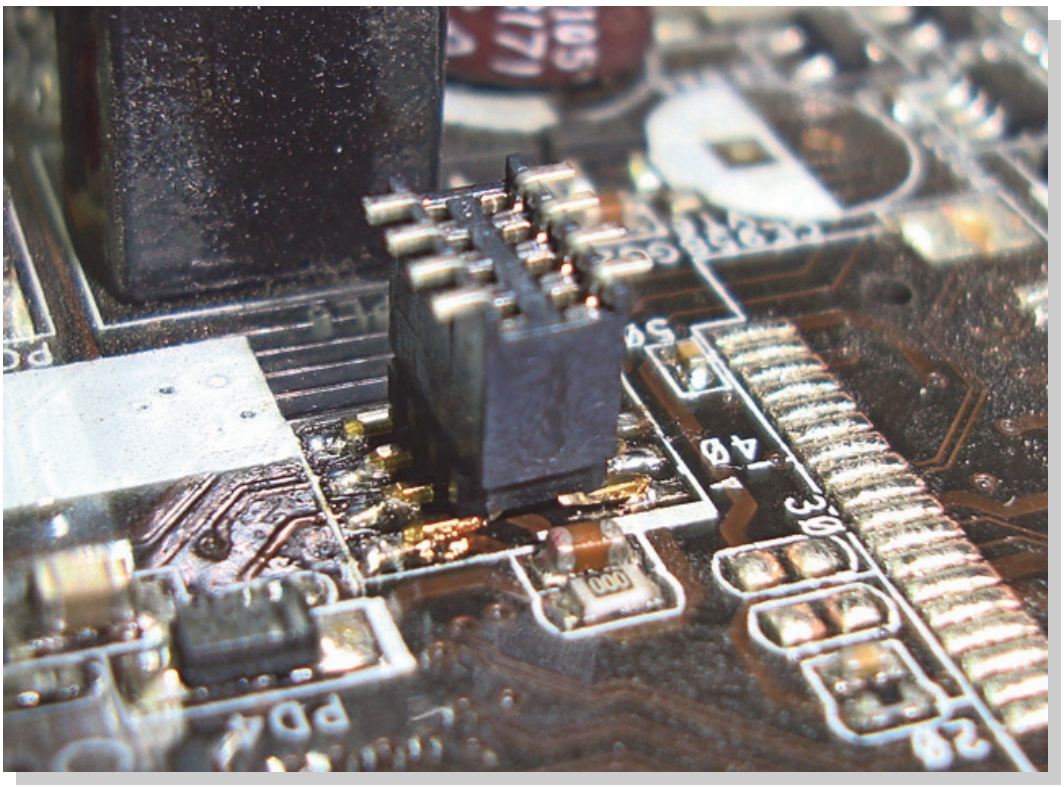
Quick instructions to connect using fixed method



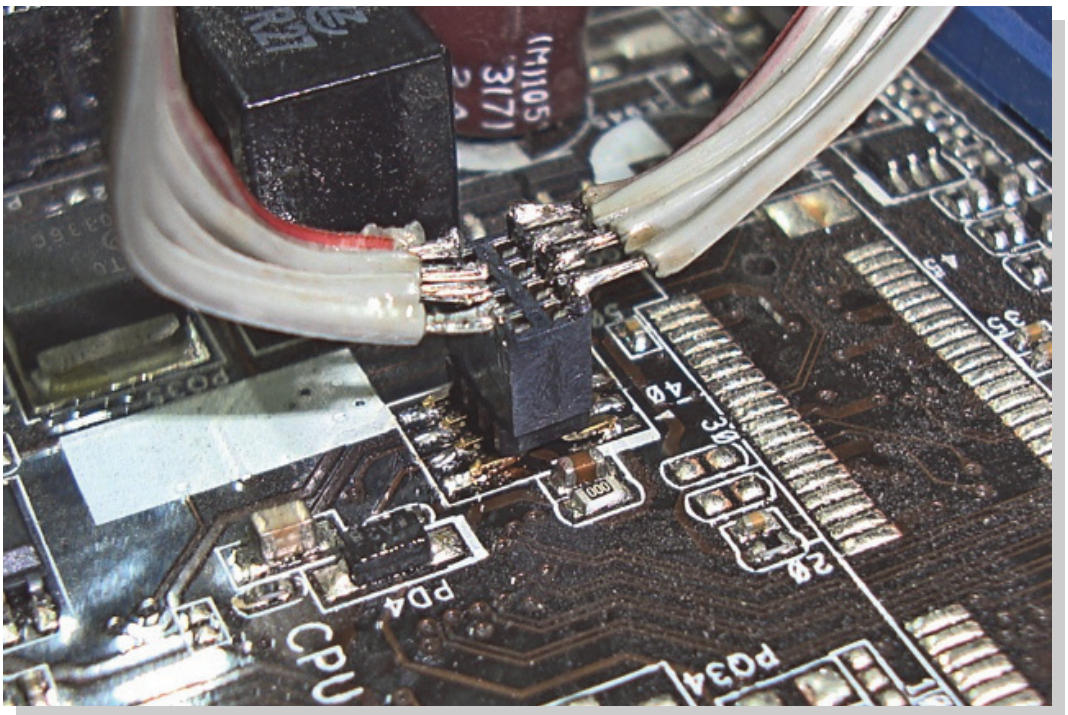
1. Make sure the board is disconnected from power source. Identify the SPI flash on the motherboard. It's usually near the south bridge and has a SO8 or DIP8 or SO16 package. If the motherboard has dual flash, identify the primary one.



2. Once identified de-solder it and solder a compatible header on it.



3. Plug in a complementary receptacle for the header.



4. The wires coming from MBD SPI port on the test interface board that are soldered to a complementary receptacle is used to make connections between the header and the test interface board. Wires should be as short as possible.

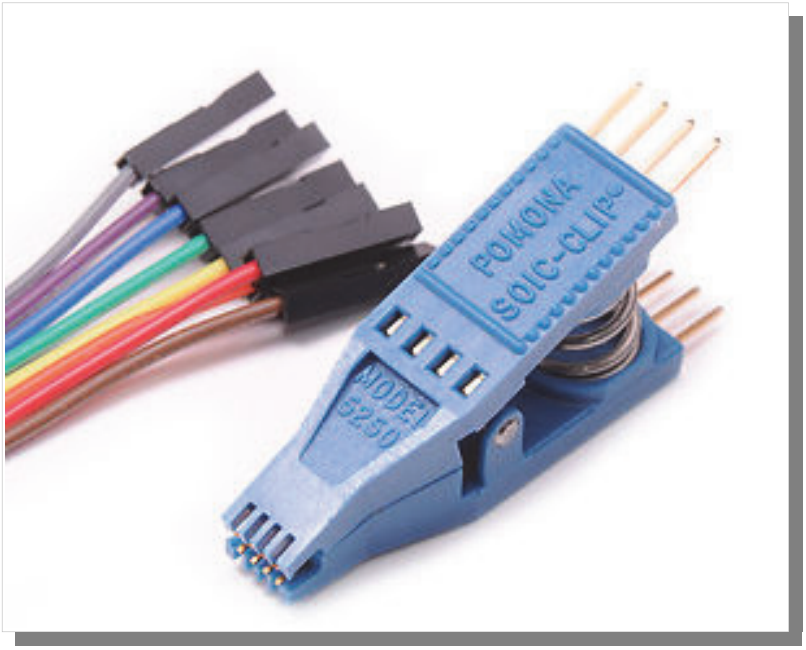
A table of recommended headers for different SPI flash IC footprints is given as follows:

Description	Generic part name	Recommended part
Header for SO8	1.27x1.27MM, SMT, Vertical, Header, 8 Pin	FCI 20021121-00008C4LF
Receptacle for SO8 Header	1.27x1.27MM, SMT, Vertical, Receptacle, 8 Pin	FCI 20021321-00008C4LF
Header for SO16	1.27x1.27MM, SMT, Vertical, Header, 16 Pin	FCI 20021121-00016C4LF
Receptacle for SO16 Header	1.27x1.27MM, SMT, Vertical, Receptacle, 16 Pin	FCI 20021321-00016C4LF

Some motherboards have a DIP8 socket. This can be even more convenient as male-female jumper wires can be used to make connections without needing to solder/de-solder anything on the motherboard.

Note: If soldering wires directly to the IC footprint please be aware that the stress from an accidental pull of the wire can peel off the solder pads and damage the motherboard irreversibly.

Test-clip method



An SO8 IC test clip from Pomona electronics

The test clip method relies on the #HOLD pin on SPI flash chips to mute its communication and make it invisible on the SPI bus. When HOLD is enabled the MISO line goes to high impedance state. This allows us to establish communication between the motherboard's south-bridge and the test flash on TIB.

Connect the MISO, MOSI, #CS, CLK, VCCP, GND pins of the flash IC using test clip to the respective pins of header P1 on TIB.

To activate hold, the #HOLD pin needs to be connected to ground since it is an active-low pin. To prepare for this method check for resistance between the #HOLD pin and VCC pin of the motherboard flash chip using a resistance meter. There can be two cases:

- Case I: It measures a high resistance in order of kilo-ohms or maybe more.

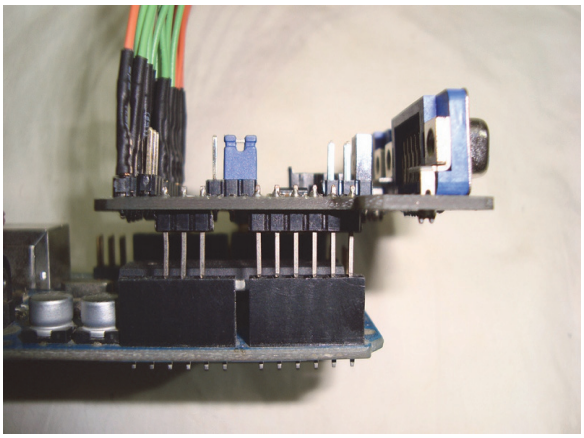
This is fine, the #HOLD pin is pulled up through a resistor and can be safely pulled-down to ground to activate it. Connect #HOLD pin to ground to activate #HOLD.

- Case II: It indicates a short circuit. You may not be able to use the test-clip method because connecting the #HOLD pin to ground will effectively create a short circuit between the motherboard's VCC and ground and this will damage the board.

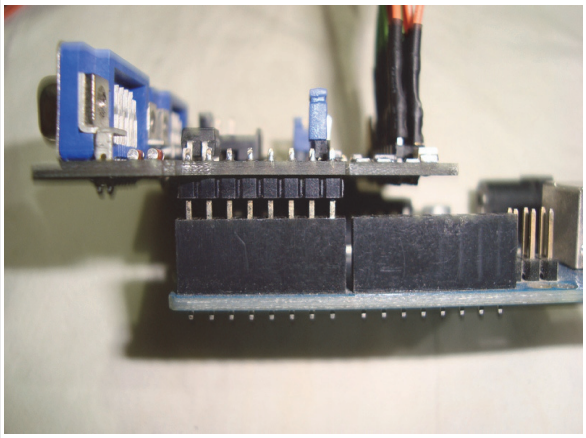
If the test-clip method is used for repeated tests over long term it is advisable to use a glue gun to put hot-melt adhesive between the board and test clip. Ensure that the glue doesn't stick on any of the components on the board.

Connecting an Arduino Uno (P6, P7, P9)

Arduino Uno, Duemilanove, Leonardo and their clones can work with this board. Arduino Uno is recommended as it comes with a unique USB serial id. USB serial id helps to uniquely identify a device when multiple of the same are connected to a computer.



Test interface board partially inserted into an Arduino in correct alignment. Shows power and analog ports.

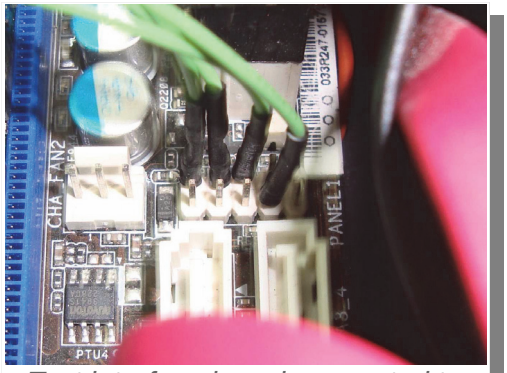


Test interface board partially inserted into an Arduino in correct alignment. Shows digital port

Take care when aligning and connecting the TIB to the Arduino. P7 of the TIB should be inserted to D0-D7 of the Arduino and P9 should be inserted to A0-A5 of the Arduino.

Connecting to motherboard front panel header (P5)

The power and reset button pins in the motherboard are inputs that have been pulled up internally. Shorting the input to ground is same as a button-press. In each pin pair for power and reset buttons, one of the pin is an input and the other pin is ground. You need to connect the TIB's output to the input pin not the ground. To identify an input pin try shorting to ground through a resistor (100 ohms to 1K) one of the pins to ground. If the button activates it's the input, otherwise it's ground. Connect these inputs to the corresponding button outputs in P5.



Test interface board connected to front panel header of a motherboard

The power LED signal is read by the TIB through an opto-coupler within the board which

is not used for opto isolation but to emulate an LED. This is needed since the LED output voltage can be less than the minimum voltage to convey a logic high, so direct connection without an opto-coupler may not work reliably.

The +ve and -ve terminals are usually marked on the motherboard's header label and in case it's not given you can try connecting either way to see which way it works.

Power LED readout is an important function. If power LED output is not available, use a voltage rail such as +5V or +12V from the ATX power supply and connect a resistor in series. The resistance required will be given by the formula:

$R = 50*V_{IN} – 60$ ohms.

(derived using the voltage divider rule and the dynamic resistance 60 ohms of the PC817 LED at operating point 1.2V, 20mA)

Connecting to Video outputs (X1, X2)

The test interface board has a VGA pass-through. The VGA output can be connected to one of the ports. The other port is optional and can be used to connect a monitor without altering the test set-up.

In case the motherboard has a different video interface such as HDMI, DVI or DisplayPort use a VGA converter. Inexpensive VGA converters are available to buy on internet.

The test interface board detects whether a VGA sync signal is present or not. In other words, it detects whether a connected monitor would have its status LED green (active) or amber (inactive) at a given time.

Command Line Interface (TIB.py)

Use the command line interface program TIB.py (python script) found in coreboot-ADTS/Test Server/Scripts to perform the test operations with the TIB. Usage details and command line options can be found in the same directory.

DIY assembly

PCBs can be ordered by sending the gerber files found in coreboot-ADTS/Test Interface Board/Kicad files/gerber/ to a PCB fabrication service.

To order components refer to the Bill of materials (BOM) found in coreboot-ADTS/Test Interface Board/Kicad files/BOM.xls

For people new to soldering:

The minimum tools required for building this is a 10-15W micro-soldering iron, good quality flux, a thin solder wire preferably with less than 0.5mm cross-section diameter and a pair of tweezers.

Melt solder to one of the pads for every component's footprint. Use tweezers to correctly place the component and then solder the terminal where the solder was melted. Now it will be easier to solder the other terminals of each component.

Disabling Arduino auto-reset

Arduino Uno auto-resets the micro-controller when a serial connection is established. This is useful for Arduino IDE as the microcontroller loads its boot-loader on reset and starts accepting the payload. This feature has side-effects when sending commands to

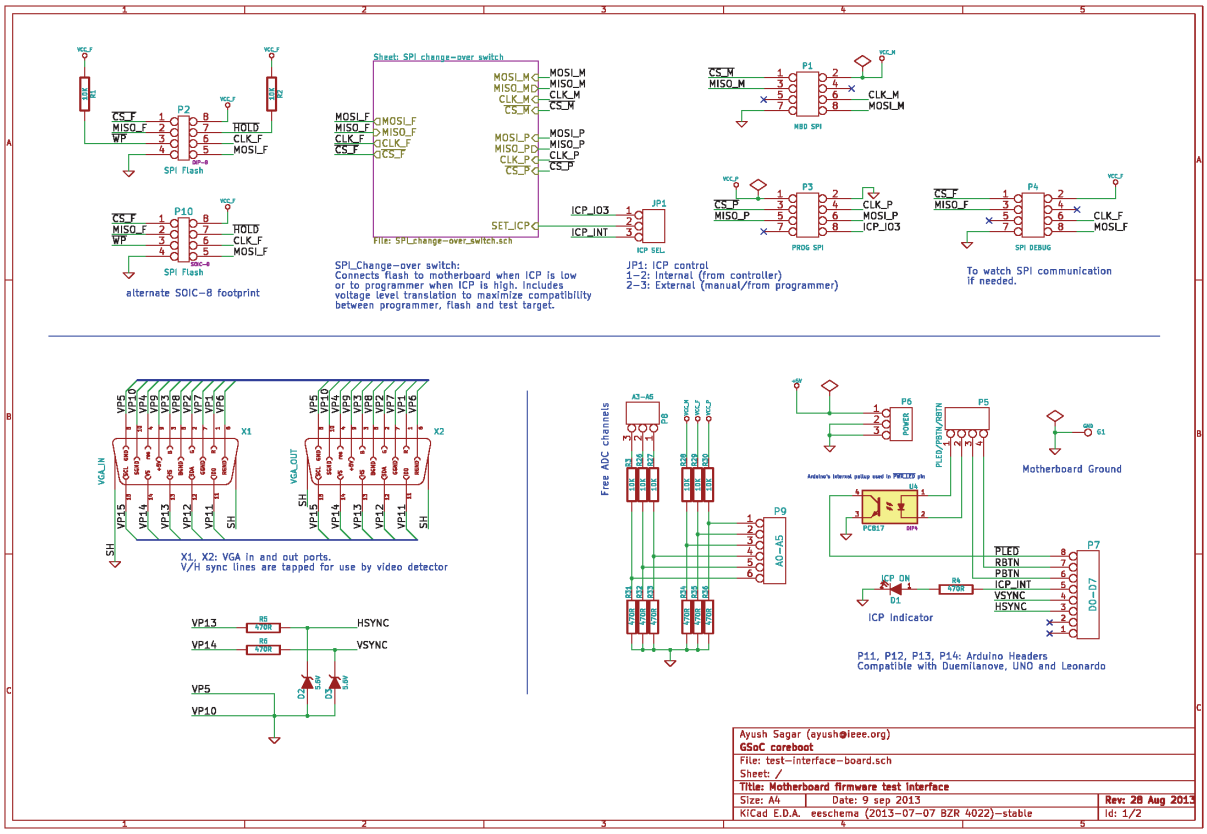
the test interface board through the scripts. Disabling auto-rest is hence desirable in our application.

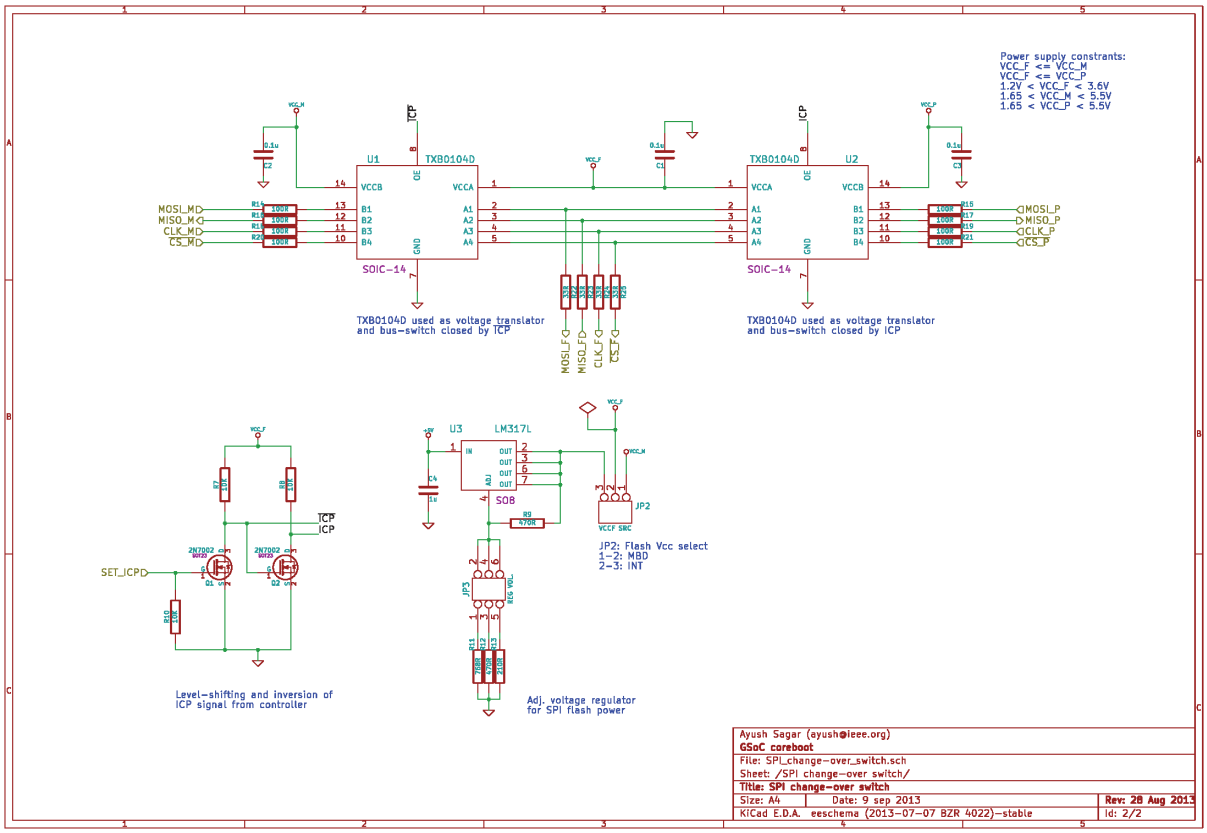
Follow instructions on http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection to disable auto-reset.

Arduino Sketch

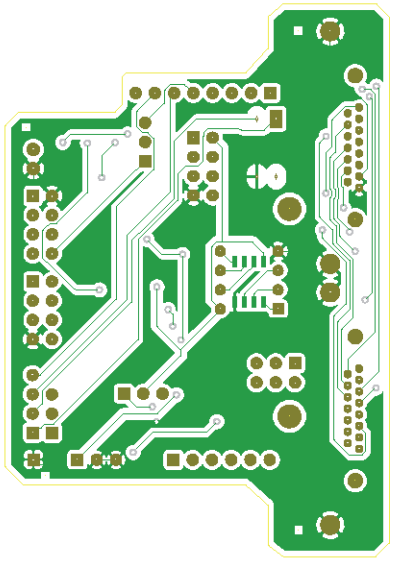
Burn the Arduino sketch coreboot-ADTS/Test Interface Board/Arduino sketch/TIB/TIB.ino to an Arduino board using the Arduino IDE. Make sure to select the correct serial port and correct board type. This was tested to work on Arduino IDE v1.0.3 as well as v1.0.5

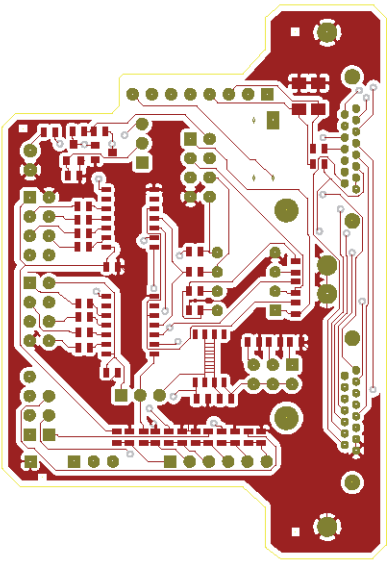
Schematics

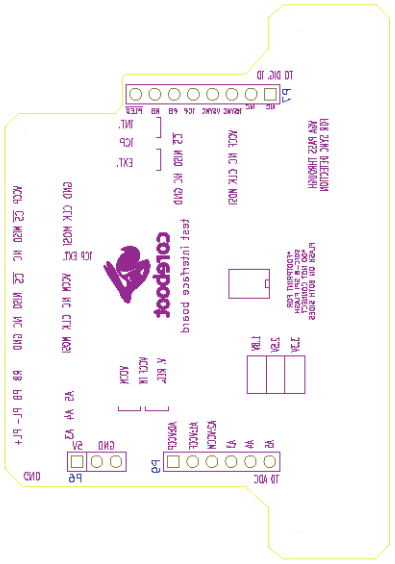

SPI Flash
SPI Flash
alternate SOIC-8 footprint
Sheet: SPI change-over switch
File: SPI_change-over_switch.sch
SPI_Change-over switch:
Connects flash to motherboard when ICP is low
or to programmer when ICP is high. Includes
voltage level translation to maximize compatibility
between programmer, flash and test target.
JP1: ICP control
1-2: Internal (from controller)
2-3: External (manual/from programmer)
MBD SPI
PROG SPI
SPI DEBUG
To watch SPI communication
if needed.
X1, X2: VGA in and out ports.
V/H sync lines are tapped for use by video detector
Free ADC channels
Motherboard Ground
ICP Indicator
P11, P12, P13, P14: Arduino Headers
Compatible with Duemilanove, UNO and Leonardo
Ayush Sagar (ayush@ieee.org)
GSoC coreboot
File: test-interface-board.sch
Sheet: /
Title: Motherboard firmware test interface
Size: A4
Date: 9 sep 2013
Rev: 28 Aug 2013
KiCad E.D.A. eeschema (2013-07-07 BZR 4022)-stable
Id: 1/2

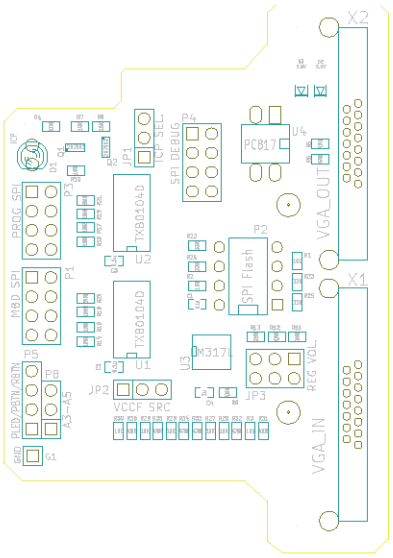

Power supply constrants:
VCC_F <= VCC_M
VCC_F <= VCC_P
1.2V < VCC_F < 3.6V
1.65 < VCC_M < 5.5V
1.65 < VCC_P < 5.5V
TXB0104D used as voltage translator
and bus-switch closed by ICP
TXB0104D used as voltage translator
and bus-switch closed by ICP
Level-shifting and inversion of
ICP signal from controller
Adj. voltage regulator
for SPI flash power
JP2: Flash Vcc select
1-2: MBD
2-3: INT
Ayush Sagar (ayush@ieee.org)
GSoC coreboot
File: SPI_change-over_switch.sch
Sheet: /SPI change-over switch/
Title: SPI change-over switch
Size: A4
Date: 9 sep 2013
Rev: 28 Aug 2013
KiCad E.D.A. eeschema (2013-07-07 BZR 4022)-stable
Id: 2/2

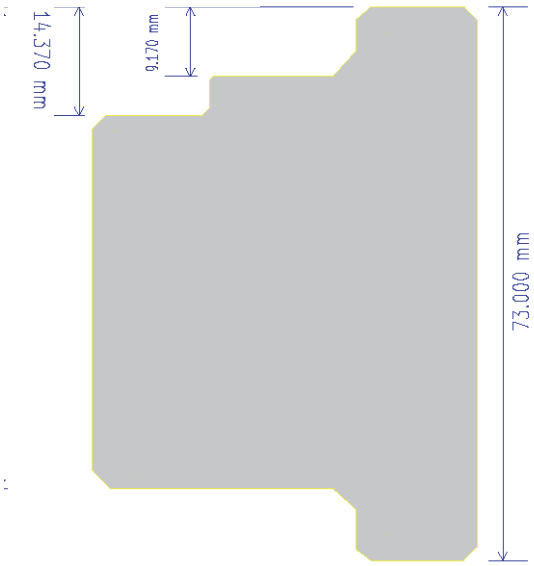
PCB layout



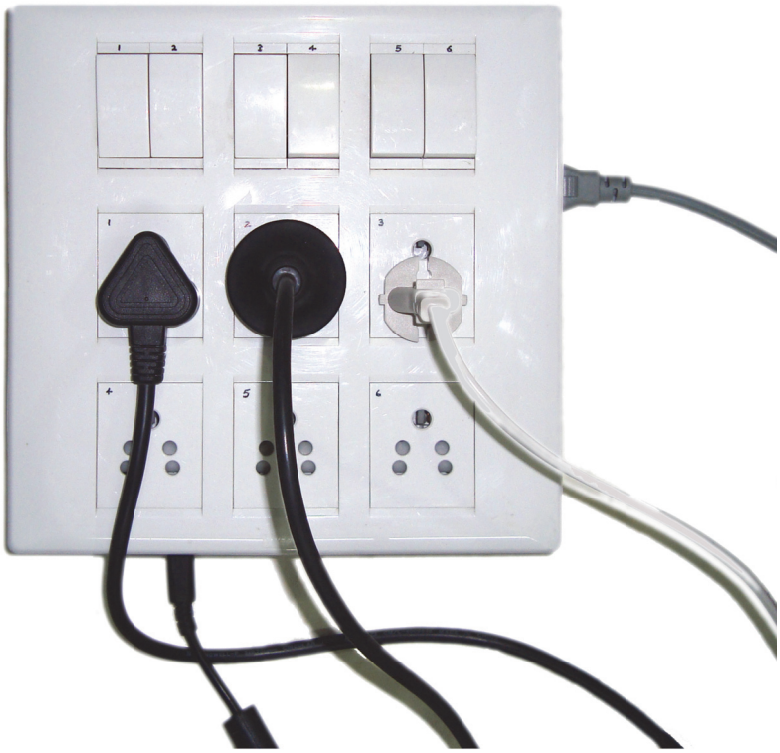



coldpcb
test interface board


X2
X1
VGA_OUT
VGA_IN
PC817
SPI Flash
PROG SPI
MBD SPI
SPI DEBUG
VCCF SRC
REG VOL
TXB0104D
TXB0104D
DM3174


14.270 mm
9.170 mm
73.000 mm

Overview



Prototype of the Coreboot USB power strip (a box actually) made with locally available parts. 220V Indian sockets used.

The test logic running on a test server would often need to switch off and on the mains supply to a system under test.

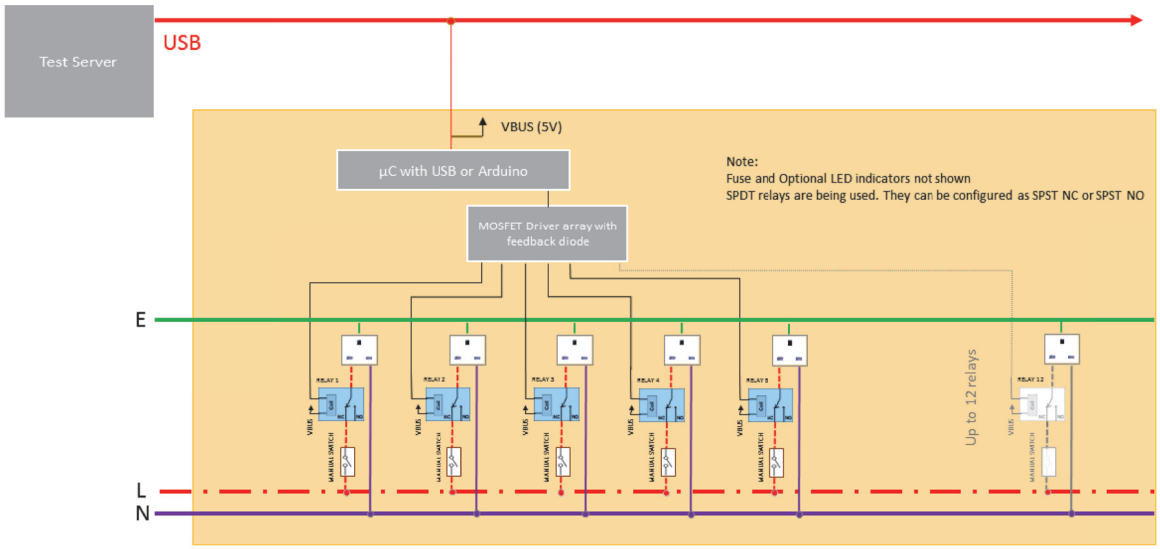
While there are some controllable power-strips available on the internet, they're not usable enough for the following reasons:

1. Non-smooth availability
2. No Linux compatible command line interface
3. No choice of no. of sockets and the quality of sockets.
4. It may not have the type of sockets for your region.

Keeping the above points in view a new controllable power strip called coreboot USB power strip has been designed that is easily constructable with locally available material, safe and has a Unix command line interface.

Operation

Hardware



General schematic of the coreboot USB power strip

The coreboot USB power strip consists of a switch box that contains manual switches and software controllable relays to control mains power to the connected loads using an Arduino Uno. Uno is preferred as it comes preprogrammed with unique serial id string that helps to uniquely identify the power strip when multiple of the same are connected to a test server. The relays are chosen such that the max power consumption of the entire circuit lies below 500mA which is the current limit for most USB ports.

Normally closed (NC) relays are connected in series with each supply so that the device stays normally on. A NC relay normally acts as a closed switch and breaks the connection when it is activated. As a result of putting the NC relay and manual switch in series with the load and mains supply:

- when disconnected from the test server, the power strip operates as a normal power strip controlled through manual switches which is an added advantage.
- the power strip doesn't abruptly shuts off the connected load when it loses USB power or communication with the test server.
- manual switches are always there to disconnect a load.

Note: For the load that needs to be controlled and operated through software, the manual switch has to be *on* otherwise the software control would have no effect.

Software

The controller based on Arduino can be controlled using powerstrip.py (python script) found in coreboot-ADTS/Test Server/scripts. Explanation for usage and command line options can be found in the same directory. The script basically activates or deactivates specified switches on the specified power-strip.

The serial communication between the test server and power strip is based on the *Modbus RTU* protocol. Modbus is the de-facto industry standard for serial communication for its robustness, simplicity and compatibility. The test server here is a Modbus master and the USB power strip is the Modbus slave. The Modbus master sees the slave as a bunch of programmable registers (bits in this case). Bits are also known as coils especially when they drive something (relay coils in this case). The relays 1 to 6 are seen as coil 1 to 6 by the Modbus master. Modbus master is implemented in powerstrip.py using minimalmodbus library found at http://minimalmodbus.sourceforge.net/ and the Modbus slave is implemented using the arduino-modbus-slave library https://code.google.com/p/arduino-modbus-slave/

Please see Modbus specs for more information about Modbus:

http://www.modbus.org/specs.php

As a result of using Modbus this power strip is compatible with other automation software (SCADA/HMI).

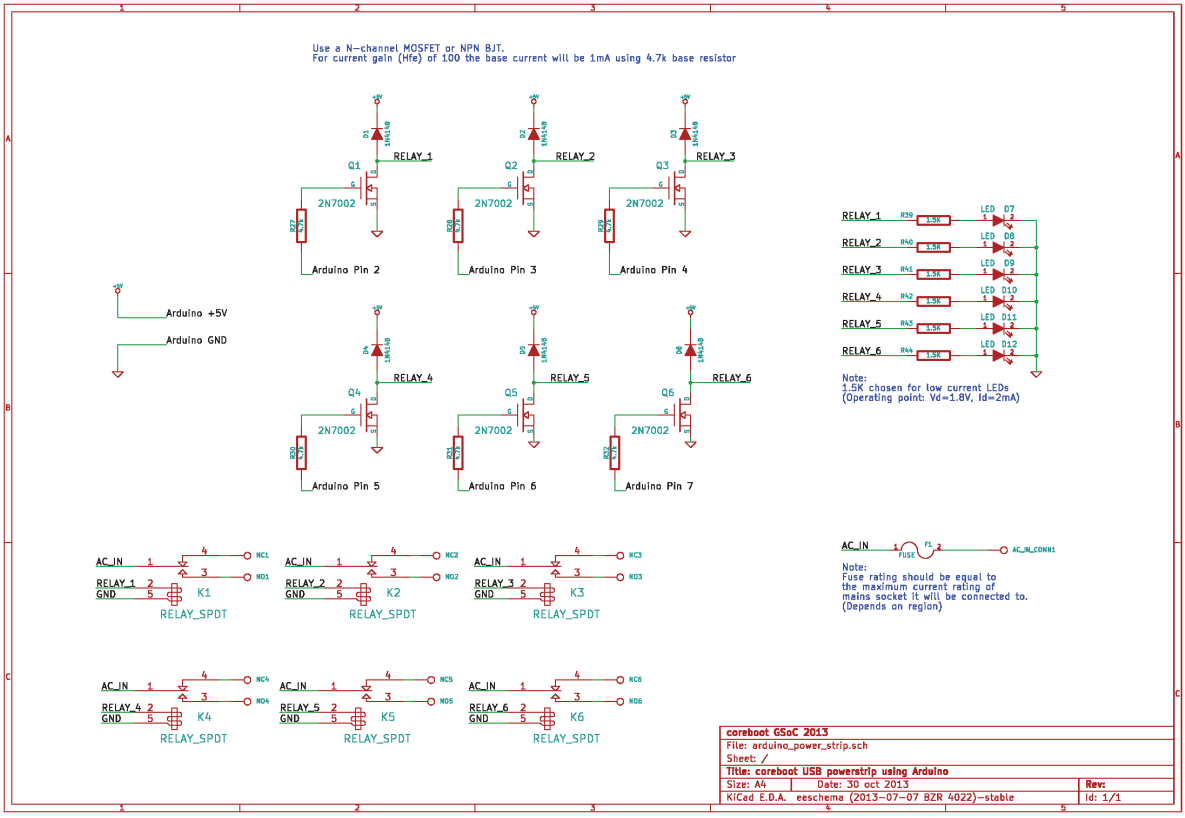
Construction

Disclaimer: By proceeding with construction and/or use of this device you understand that the device uses dangerous voltages and currents and you agree to be liable for any injury, loss of life or property, violation of law from the use or construction of this device.

Furthermore, please do not proceed with construction unless you have the necessary background in electrical circuits.

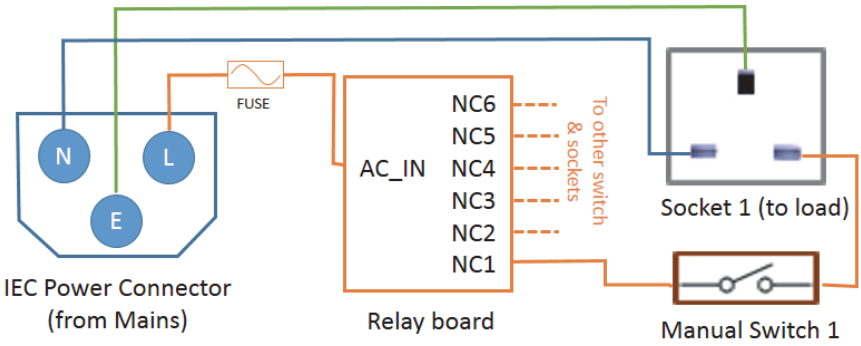
Schematics and wiring diagram

A circuit schematic for the relay board that uses 6 x SPDT 5V 72mA sugar cube relays controlled using an Arduino Uno is given as follows:



This relay board connects to the mains supply circuit through the AC_IN port which is the common AC input and NC1, NC2 .... NC6 which are the AC outputs for individual switches.

Please see the wiring diagram below to understand how the mains circuit is connected to the relay board.



It may be noted that to each switch and socket pair the Neutral (N) and Earth (E) are common and are connected directly to the mains inlet.

Sourcing parts

The following parts will be needed:

1. A Modular switch plate and PVC gang box

A modular switch plate helps in easy construction and reconfigurability. Use a compatible gang box. PVC gang box must be used instead of a metal gang box so that it can be easily modified.

Choose the switch plate and gang box considering the clearance needed to fit the relay board and Arduino beneath the switch and socket modules. At least 2.1 cm clearance is needed between the switch/socket modules and the gang box plane when the recommended relays are being used.

2. Motherboard stand-offs, and screws.

These will be needed to fix the PCBs to the back plane of the gang box.

3. Some cutting and drilling tools

Small cuts will be needed to make opening for USB connector, IEC power jack and fuse-holder. Driller might be needed to makes holes for screwing in the motherboard stand-offs. There are workarounds if such tools are not available.

4. Stranded good quality PVC insulated power wires for the high voltage circuit

Stranded conductors are needed to reduce stresses in the wire and to make good connections. Insulation should be of good quality to ensure insulation between high voltage and low voltage sides.

5. Electrical insulation tape

An insulation tape will secured wires properly to their places and cover any exposed high voltage metal for ensuring safety.

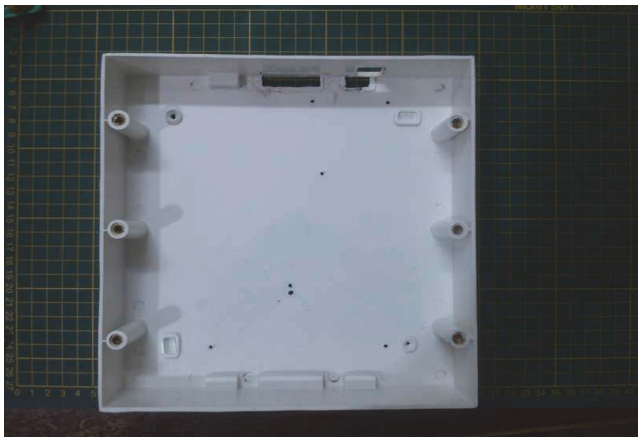
6. Circuit components

- IEC Power inlet and power cable, preferably with an integrated fuse.

- Fuse-holder, if a basic IEC power inlet without integrated fuse is used.
- 5A fuse for 220V mains or 10A fuse for 110V mains
- 6 x SPDT relays - JQC-3FC/T73 5VDC or OMRON G5LE-1 5VDC or compatible types.
- Perf-board (perforated PCBs)
- 6 x 2N7000 or BC547
- 6 x 1N4148 diode or any 1N40xx series rectifier diode
- 6x 1K or 6x 1.5K for LEDs
- Low power LEDs (2mA)
- Connecting wires and soldering tools
- Arduino Uno and USB cable

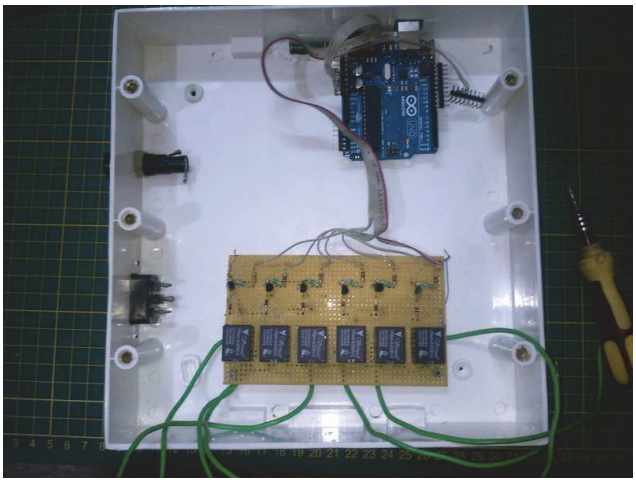
Assembly (photos from first prototype)

Get the PVC gang-box and modular switch-plate

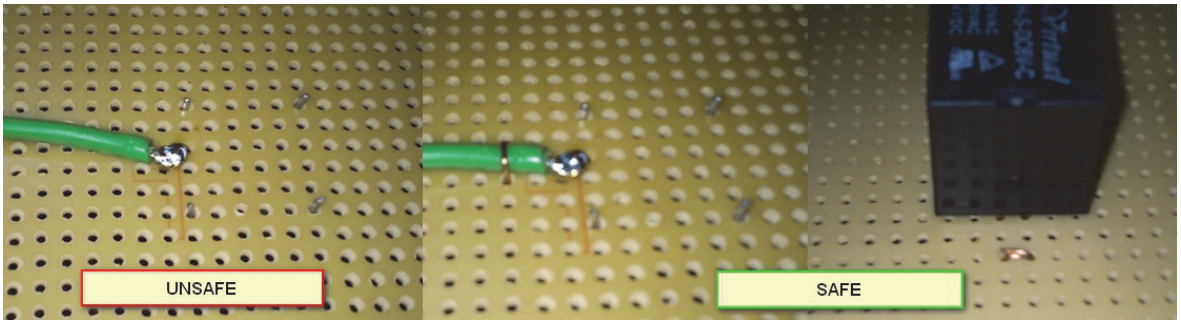


Plan positioning of the Arduino, relay board, fuse and power inlet and make cuts and holes accordingly

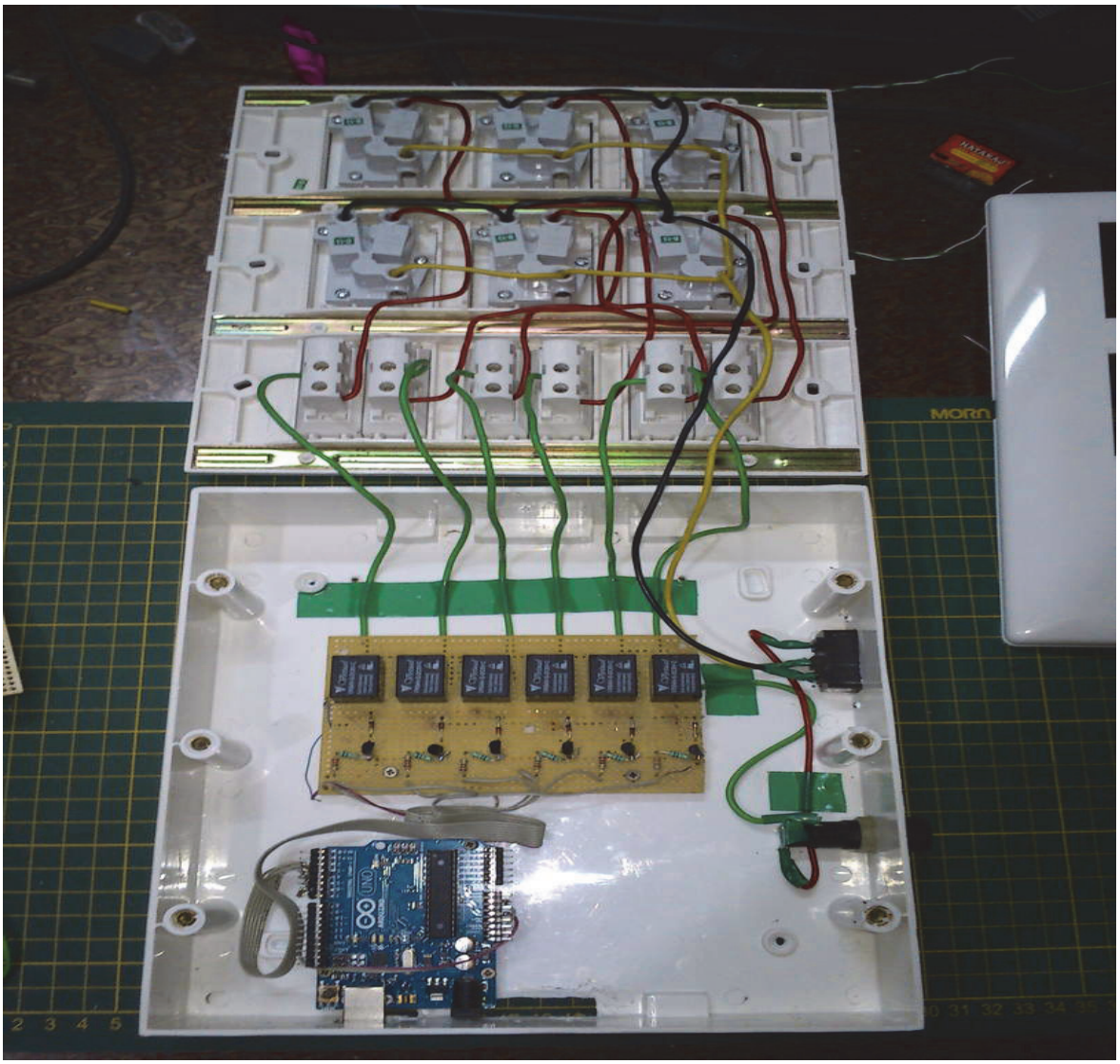
Fix the IEC power inlet, fuse-holder and motherboard stand-offs to the gang box



Start making low voltage side connections



Every wire that is being soldered to the relay MUST be fastened securely to the board because the solder joint can break due to bad soldering or melt due to overheating in case of a cold solder joint. These wires must be stable in such conditions to ensure isolation between the high voltage and low voltage circuits.



After completing all connections it is important to put electrical tape on high voltage wires to secure them in place. Putting tape on any exposed high voltage carrying conductor is also a must for safety during debugging. Keep in mind that in event of a high voltage wire melting due to short-circuit or overload, it should never come in contact with the low-voltage circuit. All high voltage and low voltage conductors must be physically separated as much as possible. Double checking the connections and check for short circuits between L, N and E of the power inlet before closing and powering on.

Disabling Arduino auto-reset

Arduino Uno auto-resets the micro-controller when a serial connection is established. This is useful for Arduino IDE as the microcontroller loads its boot-loader on reset and starts accepting the payload. This feature has side-effects when sending commands to the test interface board through the scripts. Disabling auto-rest is hence desirable in our application.

Follow instructions on http://playground.arduino.cc/Main/DisablingAutoResetOnSerialConnection to disable auto-reset.

Arduino Sketch

Burn the Arduino sketch found in coreboot-ADTS/USB Powerstrip/Arduino sketch/Powerstrip to an Arduino board using the Arduino IDE. Make sure to select the correct serial port and correct board type. This was tested to work on Arduino IDE v1.0.3 as well as v1.0.5