The Liverpool Ringing Simulator

Simulator Interface Hardware Manual

Author: Andrew Instone-Cowie
Date: 01 August 2015
Version: 1.2

Contents

Index of Figures

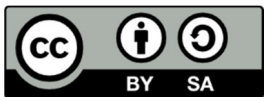
Index of Tables

Document History

Version	Author	Date	Changes
1.0	A J Instone-Cowie	25/05/2015	First Issue (Main PCB Rev B, Firmware 2.2)
1.1	A J Instone-Cowie	26/07/2015	Described cable and connector option for double detector Sensor Heads. Updated GitHub repository address.
1.2	A J Instone-Cowie	01/08/2015	Minor corrections.

Cover photograph: A completed 12-bell Simulator Interface.

Licence



[1] http://creativecommons.org/licenses/by-sa/4.0/

Introduction

This Hardware Manual describes the design and construction of the hardware for a ringing simulator interface which allows sensors, attached to real tower bells or teaching dumb bells, to be connected to a computer simulator package such as Abel[2], Beltower[3] or Virtual Belfry[4].

- Parts lists and schematics are provided. Links are also provided to suggested sources of parts, including ready-made printed circuit boards.
- Links are provided to the associated firmware source code, PCB CAD files and other supporting data hosted on GitHub.
- Details of the design and configuration of the interface firmware, and the configuration and operation of popular simulator packages, are covered in the accompanying Software Manual.
- Designs and construction details for a range of sensors are also covered separately.
- This is a "build-it-yourself" project. No pre-built hardware is available.

The hardware described in this manual is based on the approach adopted by, and seeks to maintain compatibility with, its predecessors, most notably the Bagley Multi-Bell Interface.

Licensing & Disclaimers

Documentation

[2] http://www.abelsim.co.uk/
[3] http://www.beltower.co.uk/
[4] http://www.belfryware.com/
[5] http://www.simulators.org.uk
[6] http://creativecommons.org/licenses/by-sa/4.0/

Software

All original software released as part of the Liverpool Ringing Simulator project is released under the GNU General Public Licence (GPL), Version 3[7], and carries the following disclaimers:

> *This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.*
>
> *This program 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. See the GNU General Public License for more details.*

The Liverpool Ringing Simulator Project

The Liverpool Ringing Simulator project is based on work undertaken at Liverpool Cathedral to provide a 12-bell simulator for demonstration and training purposes.

The tower of Liverpool Cathedral is open to the paying public during the day, except when ringing is taking place. During regular opening hours visitors are able to view the bells (which are usually left up) from a gallery above as they pass through the upper levels of the belfry, but have no access to the ringing room or belfry floor. These paying visitors are an important revenue stream for the Cathedral.

During ringing the tower is closed to visitors for health and safety reasons. Measurements of sound pressure taken in the belfry have shown that no reasonable level of mitigation would result in visitors' exposure to noise being reduced to a level which would be considered acceptable.

The Cathedral tower is also open weekly to the paying public for evening "Twilight Tours" during the summer months, and since 2012 this has been supplemented with access to the ringing room and belfry on one evening each month, under the supervision and escort of the Cathedral ringers. Statistic gathered by the Cathedral show that visitor attendances on the "with bells" nights were well above the average, and therefore the Cathedral were keen to continue to promote these evenings. The Cathedral ringers were also keen to use these events as a means of promoting the work of the Cathedral ringers and ringing generally to the public.

In 2013 a simulator sensor head was fitted to the Tenor bell, and this proved to be very successful. It was decided at the close of the 2013 season to explore the possibility of connecting more bells – possible as many as 12 – to a simulator. This was achieved and used successfully during the 2014 season.

Project Objectives

The initial approach adopted by the original Liverpool Cathedral Simulator project was to develop and install experimental sensors and a prototype sensor aggregation hardware interface, based on the approach adopted by David Bagley's "Multi-Bell Interface" (MBI), with the following objectives:

- To build and install a 12-bell change ringing simulator at Liverpool Cathedral for use in training ringers and in demonstrating the bells and change ringing to the public.

[7] http://www.gnu.org/licenses/gpl-3.0.en.html

- To build an experimental simulator sensor aggregation hardware interface, based, as far as possible, on off-the-shelf hardware, and with a minimum of custom electronics construction.
- To develop experimental simulator interface firmware, with additional measurement and debugging features, and to make this available to other experimenters.
- To maintain compatibility, as far as possible, with existing available simulator software packages, interface hardware and bell sensors.
- To document the design, construction and operation of the simulator installation for re-use by other experimenters.

The use of simulators has great potential in the training of new ringers, and therefore the Liverpool Ringing Simulator Project seeks to promote the installation and take-up of simulators as a teaching aid in other towers. The project presents the work undertaken at the Cathedral, and other towers, and makes it freely available for ringers to build and install simulators at relatively low cost.

Alternative Approaches

Other approaches to connecting multiple bells to a simulator software package are possible:

- For small numbers of bells, over relatively short cable distances, the approach of using RS-232 control lines as input signals, as described in the Abel and Beltower documentation, is feasible[8]. The downsides of this approach are that modern PCs usually have not more than one (if any) RS-232 serial port, limiting the number of bells which may be connected; and that multiple long cable runs are required between the simulator PC and bell sensor heads.
- The Bagley Multi-Bell Interface, complete with sensor heads and cabling, is available from David Bagley as an off-the-shelf commercial solution[9]. This approach is probably the best option for a tower wanting a professionally designed and manufactured, vendor supported simulator solution with minimal technical input. This solution is considerably more expensive than the DIY interface described in this document (and associated Sensor Heads described separately), and its designs and operating software are not published.
- A simulator sensor solution based on wireless sensors[10] is under development. As currently designed this has some disadvantages over a wired installation (such as the need to switch sensor power on and off by hand, and the need for regular sensor battery recharging), but this approach has promise for mobile use. The system is also closed source, and designs and operating software are not published.

[8] http://www.abelsim.co.uk/doc/spconn.htm
[9] http://www.ringing.demon.co.uk/abel/abelface.htm
[10] http://www.belfree.co.uk/

Acknowledgements

The Liverpool Ringing Simulator project relies extensively on work already undertaken by others, notably David Bagley (developer of the Bagley MBI), Chris Hughes and Simon Feather (developers of the Abel simulator software package), Derek Ballard (developer of the Beltower simulator software package), Doug Nichols (developer of the Virtual Belfry simulator software package), and others. Their invaluable contributions are hereby acknowledged. Sources used are referenced in the footnotes throughout.

Thanks are also owed to the Ringing Masters of Liverpool Cathedral and of St George's, Isle of Man, for their willingness to be the crash test dummies of simulator design and testing.

Simulator Interface Design

Overview

Typical Simulator Installation

The following diagram illustrates the general arrangement of a Simulator installation using a sensor aggregation hardware interface.

Multiple Sensor Heads in the belfry are connected to a hardware Simulator Interface. A single data cable transmits the aggregated signals from the Simulator Interface to the Simulator PC, using a mutually agreed protocol. The same cable feeds power from a low voltage power supply in the ringing room back up to the Simulator Interface to power both Interface and Sensor Heads.

In the ringing room, a PC runs a Simulator Software Package which interprets the received signals and turns them into the simulated sound of bells.

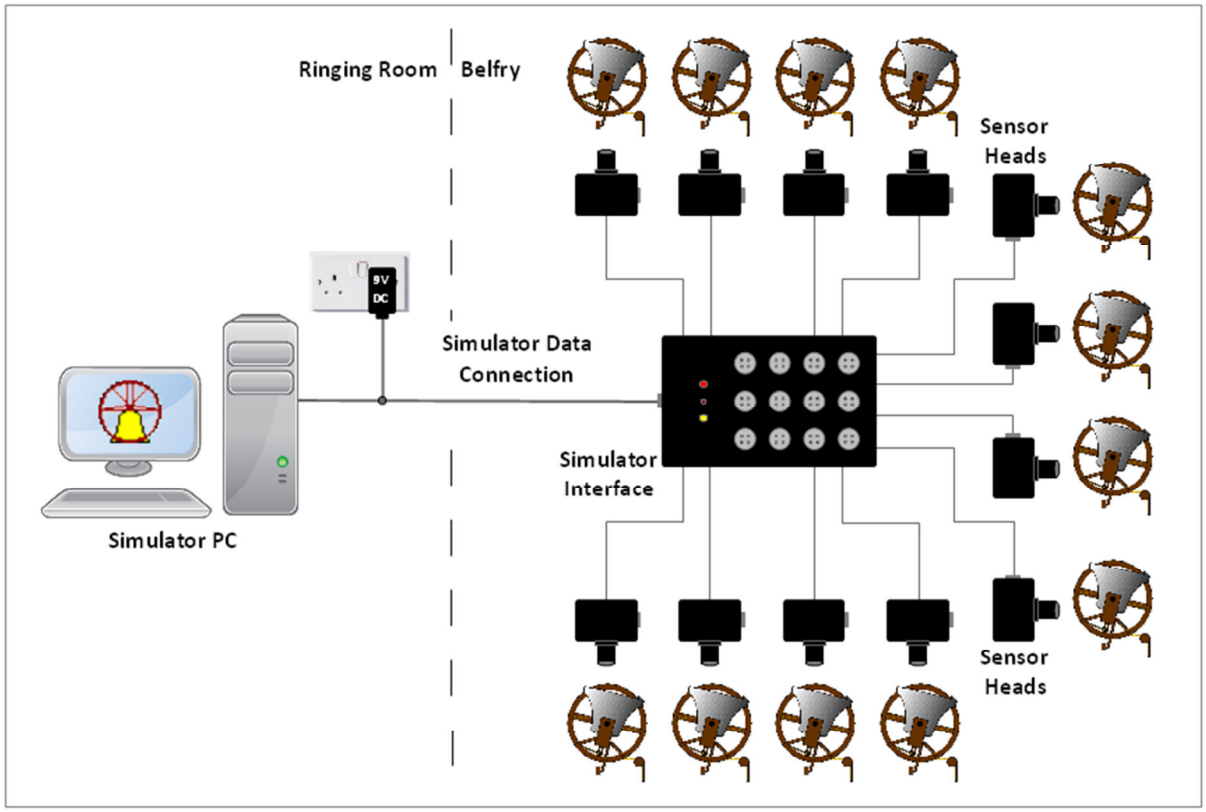


Figure 1 – Simulator General Arrangement

This manual describes the design and construction of the Simulator Interface hardware component of the Simulator.

Simulator Component Definitions

Within this document the following terms are used as defined in the table below:

Table 1 – Definitions of Terms

Term	Definition
Sensor Head	A Sensor Head detects the position of a real bell (or a dumb bell) at the bottom of its swing, and sends a signal at that time to the Simulator Interface. Multiple Sensor Heads may be installed, one for each bell to be connected to the Simulator.
Simulator Interface	The Simulator Interface aggregates the incoming signals from the Sensor Heads, and relays the signals to the Simulator PC (or to a Hardware Simulator) over a single consolidated data connection. The Simulator Interface applies a configurable delay to each signal so that it is received by the Simulator at the time the clapper of a real bell would have struck.
Simulator Data Connection	The Simulator Interface is connected to the Simulator over a single power/data connection. For compatibility with existing installations and software, the Liverpool Ringing Simulator project Simulator Interface uses a RS-232 serial data link running at 2400 bps. The cable carrying the Simulator/Power Data Connection is also used to supply power to the Simulator Interface and Sensor Heads from a low-voltage power supply located in the ringing room.
Simulator Interface Protocol	The Simulator Interface communicates with the Simulator over the Data Connection using a mutually agreed logical protocol. For compatibility with existing installations and software, the Liverpool Ringing Simulator project Simulator Interface implements the core Bagley MBI Protocol, with additional extensions for debugging and configuration purposes. These extensions are described in the accompanying Software Manual.
Simulator	Collectively, the combination of a Simulator Software Package running on a Simulator PC; or alternatively a dedicated Hardware Simulator device fulfilling the same function.
Simulator Software Package	The Simulator Software Package produces the sound of simulated change ringing through loudspeakers and/or headphones, in response to input signals from the Simulator Interface and its own internal instructions, method definitions, etc. The Liverpool Ringing Simulator project Simulator Interface has been tested with a number of popular Simulator Software Packages. These are listed below, and details of configuration can be found in the accompanying Software Manual.
Simulator PC	The Simulator PC is a general purpose computing platform which runs a Simulator Software Package, typically an Intel-compatible PC running a version of Microsoft Windows. At least one RS-232 serial port (or a USB to RS-232 adapter) is required for the Simulator Data Connection, and loudspeakers or headphones.
Hardware Simulator	A Hardware Simulator is a dedicated stand-alone proprietary device which fulfils both the hardware and software functions of a Simulator.

	An example of a Hardware Simulator is the Bagley "Ringleader" Simulator[11]. This device has not been available for new supply for some years.

Bell Sensing Method

Sensor Types

Over the years a number of different approaches have been used for detecting the position of a tied or dumb bell, and translating that position into a signal for use by a Simulator.

- Mechanical switches or contacts suffer from reliability problems, including mechanical fatigue and environmental damage (e.g. corrosion or ingress of dirt), and are not now widely used. Mechanical switches also typically suffer from a high degree of contact "bounce".
- Electro-Magnetic sensors, typically consisting of a glass-encapsulated reed switch attached to the bell frame and a small magnet attached to the wheel shroud, have also been used, but typically require very close alignment of switch and magnet. They can also suffer from fatigue and contact bounce.
- Inductive sensors, which have a fixed detector circuit attached to the bell frame, and a wire coil fitted to the wheel shroud, have also been used. In principle the absence of moving parts should make these a very reliable option. However a key component of the original design[12] is no longer available, and parts for an alternative design[13] are described as difficult to obtain.
- Aidan Hedley has produced a sensor design[14] using a Honeywell magneto-resistive sensor, activated by a small, powerful rare earth magnet mounted on the wheel shroud. A complete design for a version of this sensor is available on the Liverpool Simulator website, and a Sensor Head to this design is currently under test at Liverpool Cathedral.
- A wireless, accelerometer based sensor has been developed and prototyped as the Belfree simulator[15]. This sensor is mounted on the wheel of the bell and communicates with the Simulator PC via a radio link. At the time of writing these are at a relatively early stage of development.
- Optical sensor heads using visible light typically consist of a light source, usually a fixed red Light Emitting Diode (LED), mounted parallel to a photo-diode or photo-transistor detector, in an enclosure attached to the bell frame; and a reflector mounted on the wheel shroud. Visible light optical sensors are widely used, are very reliable, and this is the approach adopted by the Bagley MBI sensor heads[16]. Their use can be problematic in areas with high ambient light levels. Other designs include additional pulse-shaping or timing circuitry[17].
- Optical sensor heads using infra-red light are similar in general design, but have reduced susceptibility to ambient visible light levels. A design, which has been adopted as the current standard sensor for the project, for a Sensor Head using a low cost commercially available

[11] http://www.ringing.demon.co.uk/ringleader/ringleader.htm
[12] http://www.abelsim.co.uk/doc/tbellex2.htm
[13] http://www.gremlyn.plus.com/ahme/prox_sen.html
[14] http://www.gremlyn.plus.com/ahme/mag_sen.html
[15] http://belfree.co.uk/
[16] http://www.ringing.demon.co.uk/sensors/sensors.htm
[17] http://www.abelsim.co.uk/doc/tbellex1.htm

modulating infra-red detector is available on the Liverpool Simulator website[18], along with a design for a DIY sensor working on the same principle.

Single vs Dual Trigger

There are generally two approaches used by Simulators when detecting signals from Sensor Heads:

- The twin trigger approach uses two triggers (for example, optical reflectors) on the shroud of the wheel of each bell, one of which triggers at the moment that the bell would strike at handstroke, the other at the moment that the bell would strike at backstroke. The Simulator triggers the simulated sound of the bell as soon as the signal pulse is received (triggered by the reflector travelling past the sensor as the bell is on its way up to the balance), and then ignores the next signal pulse (as the first reflector travels back past the sensor as the bell is on its way down), and so on.
  This approach has the capacity to provide a very accurate representation of the striking of a bell, but equally requires very accurate placement of the triggers to match the strike points of the bell in motion. This may be difficult and time-consuming to determine.
- The second approach is to use a single trigger, which triggers as the bell passes through the bottom dead centre of its swing. A delay is then applied (either in the Simulator Interface or the Simulator Software Package) before the Simulator triggers the simulated sound of the bell. This approach is simpler to implement, and the single trigger is much easier to align accurately against the Sensor Head with the bell down. Work by John Norris[19] has shown that this approach can provide an acceptable degree of accuracy, with any errors considered too small to be detectable to the human ear.

The Liverpool Ringing Simulator project Simulator Interface adopts the single trigger approach, on the grounds of simplicity of installation and compatibility with other similar systems.

Simulator Interface Hardware

Overview

The Simulator Interface hardware consists of three main functional blocks:

- A 5V regulated power supply, which provides power for the interface electronics and for the Sensor Heads.
- A digital microcontroller, which detects incoming signals from the Sensor Heads, applies a configurable delay, and then sends aggregated signals to the Simulator PC in the form of a stream of ASCII characters.
- A RS-232 serial line driver, which converts the TTL-level signals from the microcontroller to higher voltage RS-232 signals, for transmission over longer distances to the Simulator PC.

All the components of these functional blocks are mounted on a single custom PCB. A small, optional PCB provides a mounting for diagnostic LEDs and a reset switch, and all components are mounted in a robust enclosure with standardised connectors for Sensor Heads and the Power/Data Connection.

[18] http://www.simulators.org.uk
[19] http://www.jrnorris.co.uk/strike.html

Eagle CAD and Gerber files are available in the GitHub repository, and links to a low-volume PCB fabricator are provided in the *Construction* section.

The same core hardware (and firmware) can connect to any number of Sensor Heads from one to 12. The only difference is in the number of connectors and size of enclosure required.

Hardware Options

Over recent years a number of low cost electronic prototyping platforms have been brought to the market, with a number of different architectures, and suitable for a wide range of experimental and embedded functions. These include single board computers running feature-rich operating systems, such as the Raspberry Pi and Beagle Board, and simpler microcontroller based products such as the Arduino family, PICAXE, ARM Cortex and others.

New devices and variants are released frequently.

The Arduino Uno development platform (based on the Atmel ATmega328P microcontroller[20]) was originally selected as the prototyping platform for the Liverpool Cathedral Simulator project, on the grounds of:

- Simplicity & low cost
- Established user base & wide community support
- Wide availability of additional prototyping peripheral hardware ("shields")

The use of the same microcontroller family has been continued for the Liverpool Ringing Simulator project, allowing the same well-supported and freely available tool chain to be used for code development and deployment.

Hardware Interrupts

The choice of the ATmega328P imposes an important constraint on the firmware design for the Simulator Interface, in that the microcontroller has limited support for hardware interrupt inputs. This is described in more detail in the accompanying Software Manual.

Microcontroller Hardware

The Simulator Interface is based on the same Atmel ATmega328P microcontroller as the Arduino Uno used in the first prototype interfaces. Using the same microcontroller makes it very easy to port the Simulator Interface firmware to the new hardware.

- The Simulator Interface makes use of the ATmega328P's internal 8MHz oscillator instead of an external 16MHz oscillator as found on the Arduino board. Although this is less accurate than the external oscillator, the internal oscillator is still adequate for the purposes of the Simulator Interface, and this reduces the overall component count and complexity of the interface board by omitting the crystal and load capacitors.
- Although the Simulator Interface uses the ATmega328P microcontroller and development tool chain, it does not use the Arduino boot loader program. The ATmega328P is programmed via an ICSP header using a programmer (the *Arduino-as-ISP* method is suggested, and is described in more detail in the *Construction* section of this manual.)

[20] http://www.atmel.com/devices/atmega328p.aspx

Serial Line Driver

- The Simulator Interface uses the Maxim MAX233 RS-232 serial line driver[21]. The MAX233 has the advantage over other line driver ICs of requiring no external charge pump capacitors, also reducing the overall component count.

Peripheral Hardware

Two items of peripheral hardware are included in the Simulator Interface design:

- Two diagnostic LEDs are included to provide visual status information on the activity of the Simulator Interface, because of the difficulty of using the serial interface for this purpose. These are optional, but recommended.
- A hardware reset switch is included to allow the microcontroller to be reset without having to disconnect the power feed or open the Simulator Interface enclosure. This is optional.

A design for a simple optional add-on LED/Reset PCB for these peripherals is also available.

Transient Voltage Suppression

The Simulator Interfaces uses a Transient Voltage Suppression (TVS) diode to provide a degree of protection against induced over-voltages on the power supply lines, for example resulting from local lightning strikes (similar devices are included in the Sensor Head designs).

The current interface design still lacks TVS devices on the sensor signal lines, mainly due the limited availability of suitable products in a through-hole mounting format appropriate for most DIY constructors.

Microcontroller I/O Pins

The following table shows the ATmega328P pin assignments for the Simulator Interface:

Table 2 – Microcontroller I/O Pin Assignments

Arduino I/O Pin[22]	ATmega328P Physical Pin	Function
0	2	Serial Port Receive
1	3	Serial Port Transmit
2	4	(Spare)
3	5	(Spare)
4	6	(Spare)
5	11	CRO Timing Pin (enabled in development code only)
6	12	Yellow Diagnostic LED
7	13	Red Diagnostic LED
8	14	Bell #1 Sensor Head (B1)
9	15	Bell #2 Sensor Head (B2)
10	16	Bell #3 Sensor Head (B3)
11	17	Bell #4 Sensor Head (B4)
12	18	Bell #5 Sensor Head (B5)

[21] http://www.maximintegrated.com/en/products/interface/transceivers/MAX233.html
[22] Pin numbers as referenced in the Arduino programming environment.

13	19	Bell #6 Sensor Head (B6)
14 (A0)	23	Bell #7 Sensor Head (B7)
15 (A1)	24	Bell #8 Sensor Head (B8)
16 (A2)	25	Bell #9 Sensor Head (B9)
17 (A3)	26	Bell #10 Sensor Head (B0)
18 (A4)	27	Bell #11 Sensor Head (BE)
19 (A5)	28	Bell #12 Sensor Head (BT)

Microcontroller Fuse Settings

The ATmega328P microcontroller has a number of configuration registers known as fuses. These are used to control the behaviour of the microcontroller[23]. Fuse values are contained in the file *boards.txt* described in the section on uploading firmware below, and are only written during the *burn bootloader* phase of programming.

Warning: Setting incorrect fuse values can render the microcontroller unusable or prevent further reprogramming without specialist hardware.

The fuse values for the Simulator Interface are defined in the following table.

Table 3 – ATmega328P Fuse Settings

Fuse	Value	Notes
Low	0xE2	8MHz internal oscillator, maximum clock startup delay.
High	0XD7	Enable serial programming, preserve EEPROM contents, boot reset vector disabled.
Extended	0x07	Brown-out detection disabled.

For more details of these fuse settings and their functions, refer to the ATmega328P data sheet, or see the online fuse calculator at:
http://eleccelerator.com/fusecalc/fusecalc.php?chip=atmega328p&LOW=E2&HIGH=D7.

Power Supply

DC power is supplied to the Simulator Interface over the Power/Data Cable at a higher voltage than required, to overcome the effects of losses due the resistance of the Power/Data Cable. Incoming power is fed to a conventional linear 5V regulator on the Simulator Interface PCB. An alternative to a conventional linear regulator is described in the Construction section of this manual, for installations with higher current requirements.

Typical & Peak Current Requirements

The Simulator Interface main board, peripheral LED/Reset board and Sensor Heads all operate at 5V DC. The typical total power requirement with a full set of 12 standard infra-red optical Sensor Heads was estimated at approximately 314mA, with no Sensor Heads triggered. However the Sensor Heads draw more current when in the triggered state, mainly because the infra-red detectors used include a small internal status LED). If the Simulator Interface is powered up when all the bells are down (and hence all the Sensor Heads are in the triggered state), the estimated peak power requirement rises to approximately 400mA. The current drawn by the Sensor Heads also varies slightly with

[23] Fuse tutorial: http://www.martyncurrey.com/arduino-atmega-328p-fuse-settings/

sensitivity, and the actual peak supply current on a prototype Simulator Interface, with a full complement of 12 standard Sensor Heads all in the triggered state, was measured at 440mA. This is well below the 1A maximum current rating of the 5V voltage regulator, however heatsinking is required as described below.

Microcontroller Package Ratings

The Atmel ATmega328P microcontroller has a source/sink current limit of 40mA per pin, and a total device package current limit of 200mA. The majority of the microcontroller pins will be connected to Sensor Head inputs, and these were measured to sink approximately 130 µA each when triggered, so neither of the microcontroller limits should present a problem. Other pins drive the diagnostic LEDs and the RS-232 line driver, and in total these do not draw more than a few milliamps.

Voltage Regulator

The Simulator Interface uses a standard linear voltage regulator to provide a regulated 5V DC supply. The voltage regulator requires an input supply of at least 7V DC in order to maintain a stable 5V DC output. The voltage regulator is fed via a polarity protection diode, which drops approximately 0.7V, giving a minimum input voltage requirement of 7.7V at the Simulator Interface input.

Cable Voltage Drop

The cable suggested for the main Simulator Power/Data Connection has a specified maximum resistance of 92Ω/km/core, or 0.092Ω /m/core[24]. The actual resistance of the (two) power supply cores of the 24m Simulator Data Connection cable used at Liverpool Cathedral was found to be 4.08Ω, or 0.085Ω/m/core, within the stated specification.

As an example, at an actual peak supply current of 440mA, this results in a voltage drop of 1.8V along the 24m of cable, and therefore a theoretical minimum supply voltage to the cable of 9.5V in order for the voltage regulator to maintain a stable 5V DC output. In practice a multi-voltage DC plug-in power supply unit rated at 1000mA with the output set to 9V has been found adequate for a cable runs up to 24m.

For exceptionally long cable runs a 12V DC supply could be used, but there is no advantage to doing so for shorter runs, and the heat dissipation of the voltage regulator will increase, with a requirement for additional heatsinking.

Heat Dissipation

At peak load, with a full complement of 12 standard sensors, the voltage regulator would be dissipating a minimum[25] of approximately 1W. The dissipation will be higher with shorter cable runs or a higher supply voltage. The data sheet for the voltage regulator indicates that this will require a heatsink.

An alternative approach, described in the Construction section of this manual, is to replace the linear voltage regulator with a switched buck regulator. These are available as drop-in replacements for the standard TO-220 regulator package, and eliminate the need for a heatsink altogether.

[24] http://www.rapidonline.com/pdf/02-0260.pdf
[25] Assuming the minimum required input voltage of 7V at the voltage regulator.

Fusing

A fuse rated at 800mA has been included in the design of the experimental Simulator Interface.

Compatibility

Hardware Compatibility

The Liverpool Ringing Simulator project Simulator Interface has been designed to be compatible, as far as possible, with existing simulator hardware, to allow operation with unmodified Simulator Software Packages which support the MBI protocol. This is described in more detail in the accompanying Software Manual. The following hardware design decisions have been made to maximise compatibility and interoperability:

- The Simulator Data Connection uses a RS-232 data link to the Simulator, running at 2400 bps, 8 data bits, 1 stop bit, no parity.
- The Sensor Head interfaces of the Simulator Interface operate at 5V DC TTL levels. Under normal (un-triggered) conditions the interface expects that the Sensor Head output pin will be HIGH (nominally +5V), and that when triggered the Sensor Head output pin will drop LOW (nominally 0V) for the duration of the trigger pulse.
- The Sensor Head interfaces of the Simulator Interface use the same 4-pin GX16-4 panel mount connectors as the Bagley MBI, with the same wiring convention. This is described below, with an optional extension for double detector Sensor Heads.

The Liverpool Ringing Simulator project Simulator Interface has been tested successfully with a standard Bagley optical Sensor Head, and a Bagley SBI[26] has been tested successfully with a project standard infra-red optical Sensor Head (described separately), and both were found to be compatible.

The Simulator Data Connection connector of the Simulator Interface uses a 5- way locking GX16-5 connector, although constructors are free to adopt whatever connector standards they wish. The use of physically compatible connectors with incompatible wiring conventions is strongly discouraged.

Simulator Software Compatibility

Tested Configurations

The Liverpool Ringing Simulator project Simulator Interface has been tested successfully with the following Simulator Software Packages:

- Abel 3.9.0b
- Beltower 2015 (12.29)
- Virtual Belfry[27] 3.1b

This is described in more detail in the accompanying Software Manual.

[26] Single Bell Interface; a single-input version of the MBI.
[27] 3.1b is the minimum version of Virtual Belfry required for proper handling of interfaces of this type.

Connector Pin-Outs

Sensor Head Connectors

The Simulator Interface uses GX16-4 (also known as “aviation”) connectors for the cabling connection to the Sensor Heads.

The wiring of this connector is shown in the following diagram:

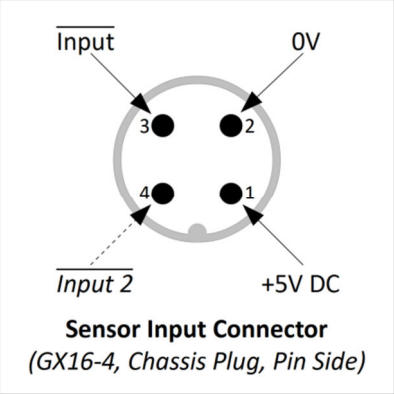


Figure 2 – Interface Sensor Head Connector

In some installations cabling may be simplified by mounting two detectors in a single sensor head, for example where the Sensor Head is mounted between two adjacent wheels. Pin 4 of the GX16-4 connector may optionally be used for the input from a second detector.

Simulator Power/Data Connector

The Simulator Interface uses a GX16-5 (also known as “aviation”) connector for the cabling connection to the Simulator.

The wiring of this connector is shown in the following diagram:

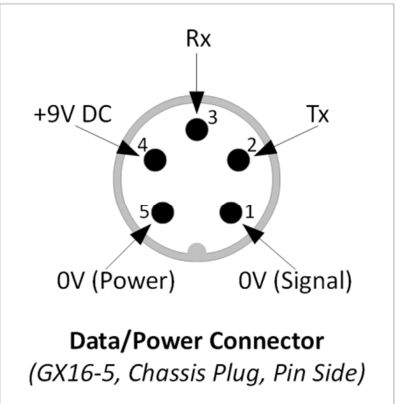


Figure 3 – Interface Data Connector

Construction

The following sections describe the construction of the Simulator Interface.

Simulator Interface Board

The Simulator Interface Board contains the power supply for the interface and Sensor Heads, the microcontroller, a RS-232 serial line driver, plus connections for diagnostic LEDs, an optional reset switch, and an ICSP[28] programming interface for firmware upload.

Eagle CAD and Gerber files for the board can be downloaded from GitHub, and ready-made boards can be obtained from OSH Park.

- https://github.com/Simulators/simulator/tree/master/hardware/simulatorinterface
- https://oshpark.com/shared_projects/lvqMwHII

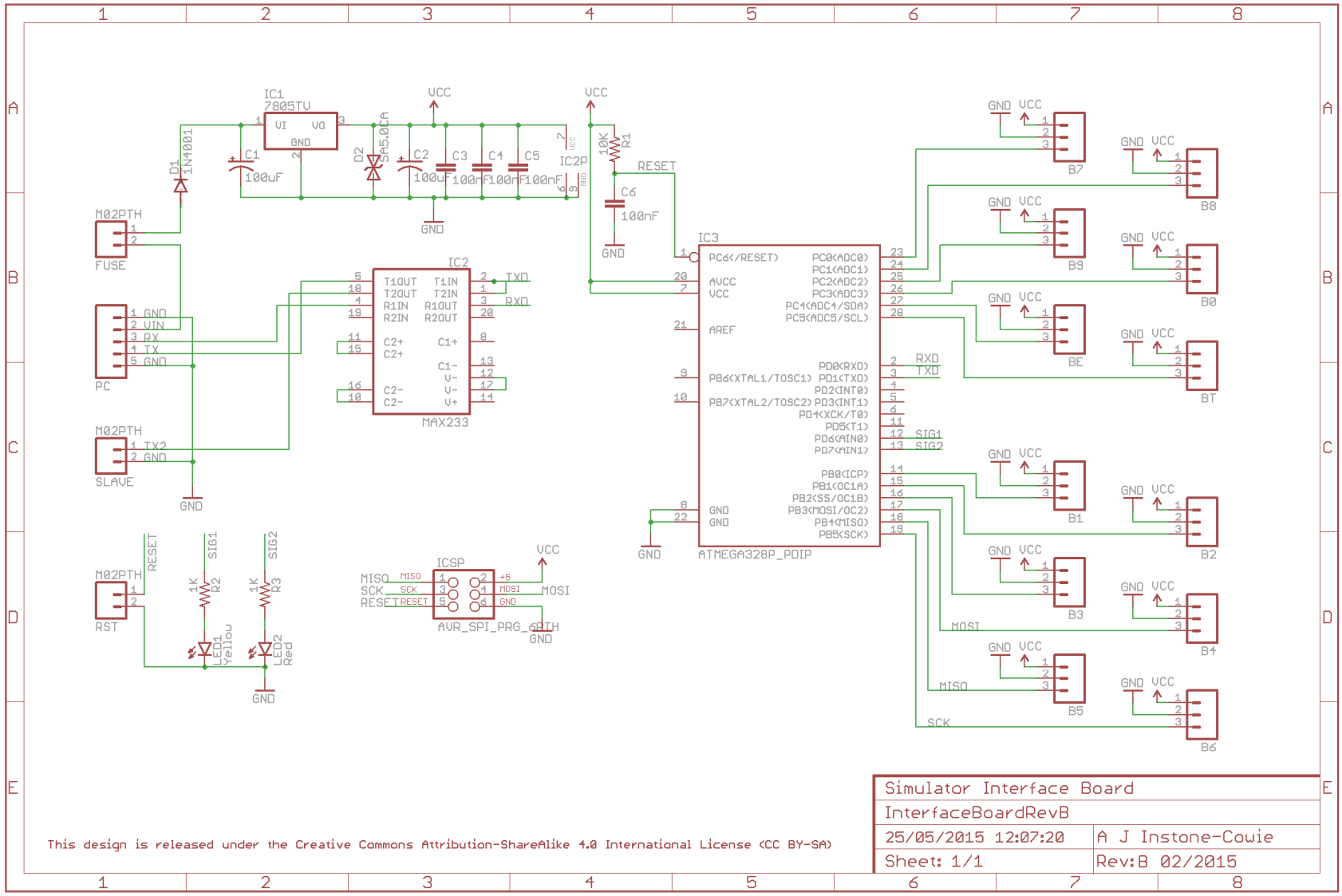
Parts List

Table 4 – Simulator Interface PCB Parts List

Reference	Component	Notes
PCB	Simulator Interface PCB – Rev B	https://oshpark.com/shared_projects/lvqMwHII
R1	10kΩ 0.6W Metal Film	
R2, R3	1kΩ 0.6W Metal Film	
C1, C2	100µF 25V Electrolytic (5mm Radial)	
C3, C4, C5, C6	100nF 50V MLCC (2.54mm Radial)	
D1	1N4001	
D2	SA5.0 CA	Transient voltage suppression diode
IC1	LM7805 & Heatsink	Alternative: Tracopower TSR 1-2450 – see text
IC2	MAX233EPP	
IC3	ATmega328P-PU	Alternative: ATmega328-PU – see text
LED1	3mm LED Yellow	Optional – see text
LED2	3mm LED Red	Optional – see text
Fuse Header	2-Pin 0.1” Male Header	
PC Header	5-Pin 0.1” Male Header	
Reset Header	2-Pin 0.1” Male Header	Optional – see text
ICSP Header	2x3-pin 0.1” Male Header	
B1, B2, B3, B4, B5, B6, B7, B8, B9, B0, BE, BT	3-Pin 0.1” Male Header	Quantity as required (1 – 12)
LED Header	4-Pin 0.1” Male Header	Optional – see text
IC Socket	20-pin, 0.3” pitch	IC2
IC Socket	28-pin, 0.3” pitch	IC3

[28] In-Circuit Serial Programming

Schematic


IC1
7805TV
VI
VO
GND
VCC
D1
1N4001
D2
SA5.0CA
C1
100uF
C2
100uF
C3
100nF
C4
100nF
C5
100nF
IC2P
10K
R1
RESET
C6
100nF
IC3
PC6(/RESET)
AVCC
VCC
AREF
PB6(XTAL1/TOSC1)
PB7(XTAL2/TOSC2)
GND
GND
PC0(ADC0)
PC1(ADC1)
PC2(ADC2)
PC3(ADC3)
PC4(ADC4/SDA)
PC5(ADC5/SCL)
PD0(RXD)
PD1(TXD)
PD2(INT0)
PD3(INT1)
PD4(XCK/T0)
PD5(T1)
PD6(AIN0)
PD7(AIN1)
PB0(ICP)
PB1(OC1A)
PB2(SS/OC1B)
PB3(MOSI/OC2)
PB4(MISO)
PB5(SCK)
ATMEGA328P_PDIP
RXD
TXD
SIG1
SIG2
MOSI
MISO
SCK
IC2
T1OUT
T2OUT
R1IN
R2IN
T1IN
T2IN
R1OUT
R2OUT
C2+
C2+
C1+
C1-
V-
C2-
C2-
V+
MAX233
TXD
RXD
M02PTH
FUSE
PC
1 GND
2 VIN
3 RX
4 TX
5 GND
M02PTH
SLAVE
1 TX2
2 GND
GND
M02PTH
RST
RESET
SIG1
SIG2
1K
R2
1K
R3
LED1
Yellow
LED2
Red
GND
ICSP
MISO
SCK
RESET
+5
MOSI
GND
AVR_SPI_PRG_6PTH
GND
VCC
MOSI
B7
B8
B9
B0
BE
BT
B1
B2
B3
B4
B5
B6
Simulator Interface Board
InterfaceBoardRevB
25/05/2015 12:07:20
A J Instone-Cowie
Sheet: 1/1
Rev:B 02/2015
This design is released under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA)

PCB Layout

The following diagram shows the layout of the Simulator Interface PCB. All components are mounted on the top (silkscreen) side of the board.

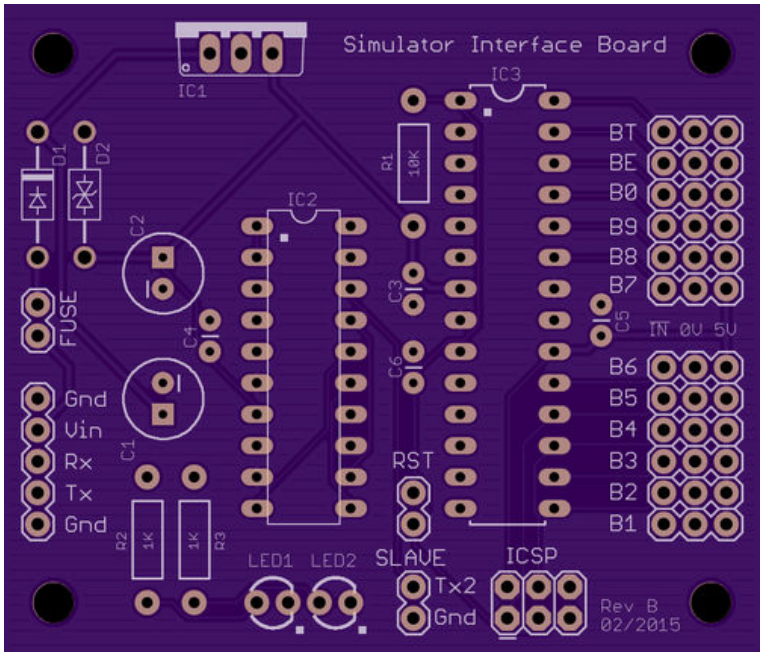


Figure 4 – Simulator Interface Board Layout

Construction

All the components on the Simulator Interface Board are mounted on top, silkscreen, side of the board.

- Start by soldering the components with the lowest profile (resistors, ceramic capacitors), then the remainder of the components in order of increasing height, ending with the header pins and voltage regulator.
- The use of IC sockets for IC2 & IC3 is optional, but strongly recommended.
- Fitting the two diagnostic LEDs is optional, but strongly recommended. The PCB design allows for the LEDs to be fitted directly to the board, or replaced by a 4-pin header for connection to off-board LEDs, or to the LED/Reset PCB described below.
- Fitting a remote reset switch is optional. If a reset switch is not required then the reset header pins (RST) may be omitted from the board.
- Fit as many 3-pin sensor input headers as required for the number of sensors to be connected to the simulator, always starting with position B1 and working upwards. The remaining sensor header pins may be omitted.
- The pair of pins marked Slave are reserved for future use.
- For high current installations, i.e. those with large numbers of sensors and/or very short power/data cable runs, consider replacing the LM7805 linear regulator with a switched buck regulator such as the Tracopower TSR 1-2450. This is a direct drop-in replacement for the standard TO-220 package regulator. The buck regulator is much more efficient than the linear version, and eliminates the need for a regulator heatsink entirely.

- Either the ATmega328P-PU or ATmega328-PU microcontroller may be used. The ATmega328P-PU low-power version is used in the Arduino Uno board, but there is no real requirement for the low-power features of this variant in the Simulator Interface, so the slightly cheaper standard version may be used instead. Note that the firmware upload procedure will require a small adjustment if the alternative microcontroller is used; this is described in the section on uploading firmware below.
- A heatsink will be required for the LM7805 voltage regulator. The size of heatsink required will depend on the number and type of sensors connected, and on the length of the main powe/data cable. The heatsink shown in the photograph below is probably rather too small for a 12-bell installation; in this case consider using the buck regulator instead.
- Before fitting the socketed ICs, connect the board to a power supply and check that +5V and 0V appear on the correct pins of the sockets and on the correct header pins. Disconnect the power supply and fit the ICs.
- If the board is powered up at this point, there will be no indication from the diagnostic LEDs. This is normal if the firmware has not yet been uploaded to the microcontroller.

A completed Simulator Interface PCB is shown in the following photograph. Note that the jumper on the fuse pins is for testing only.

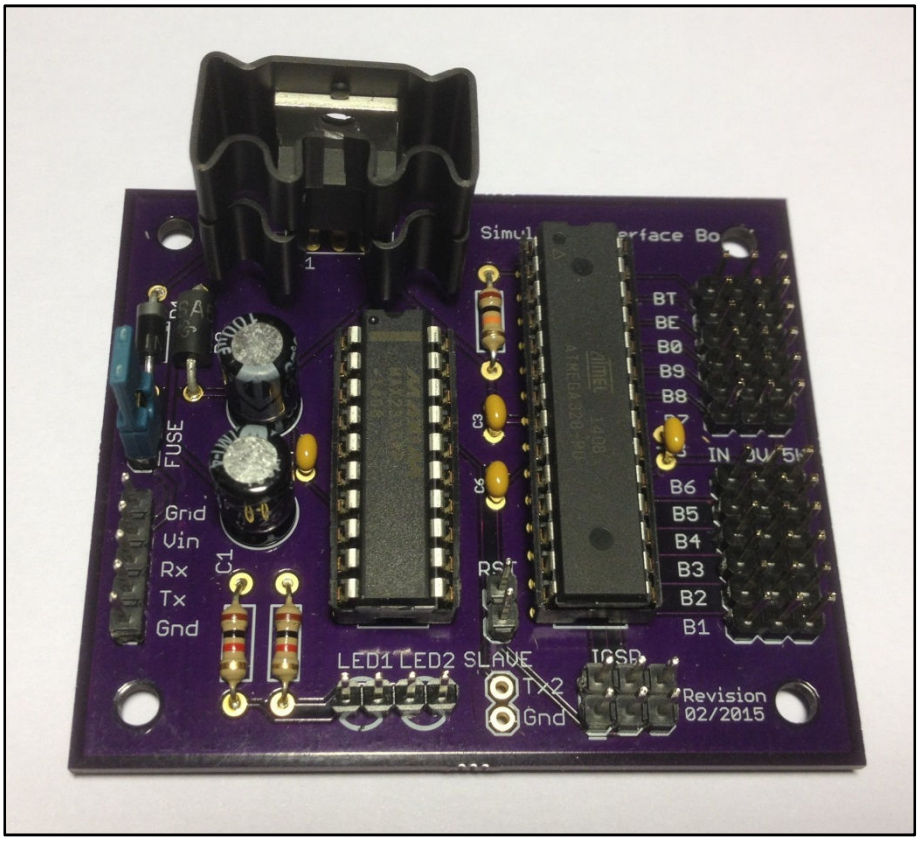


Figure 5 – Completed Simulator Interface PCB

LED/Reset Board

The optional LED/Reset board provides a mounting for the interface diagnostic LEDs, and an optional microcontroller reset switch. Other arrangements are possible for mounting these components.

Eagle CAD and Gerber files for the board can be downloaded from GitHub, and ready-made boards can be obtained from OSH Park.

- https://github.com/Simulators/simulator/tree/master/hardware/ledresetboard
- https://oshpark.com/shared_projects/9CAo7drd

Parts List

Table 5 – LED/Reset Board Parts List

Reference	Component	Notes
PCB	LED/Reset Board – Rev A	https://oshpark.com/shared_projects/9CAo7drd
LED1	3mm LED Yellow	
LED2	3mm LED Red	
SW1	Omron B3F-1070 Push Switch	Optional
Header	4-Pin 0.1” Male Header	Interface board connection

Schematic

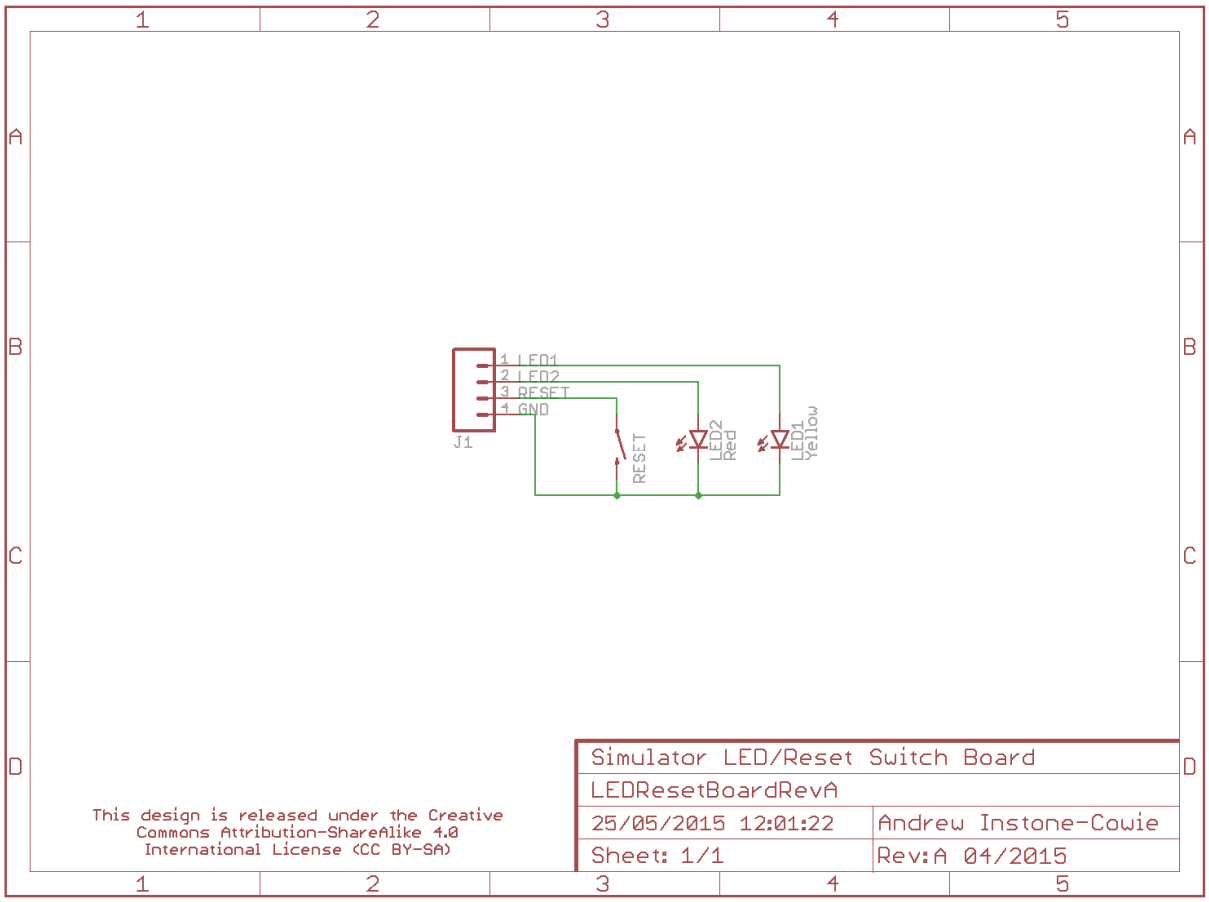


Figure 6 – LED/Reset Board Schematic

PCB Layout

The following diagram shows the layout of the LED/Reset Board PCB.

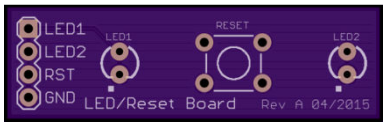


Figure 7 – LED/Reset Board Layout

Construction

The LEDs and optional reset switch (if used) are mounted on the bottom of the board (i.e. the opposite side of the board to the silkscreen text). The header pins are mounted on the top, silkscreen, side.

- If a reset switch is to be used, solder this to the LED/Reset Board first, mounting the switch on the bottom (non-silkscreen) side of the board.
- Then solder the LEDs to the same side of the board, making the sure that the flanges of the LEDs are level with the top face of the body of the reset switch, so that the switch and LEDs will sit flush against the inside of the enclosure. A simple wooden jig may be useful for aligning and levelling the components, drilled with two outer 3mm and a central 4.5mm diameter hole, on 12.7mm (½ inch) centres.

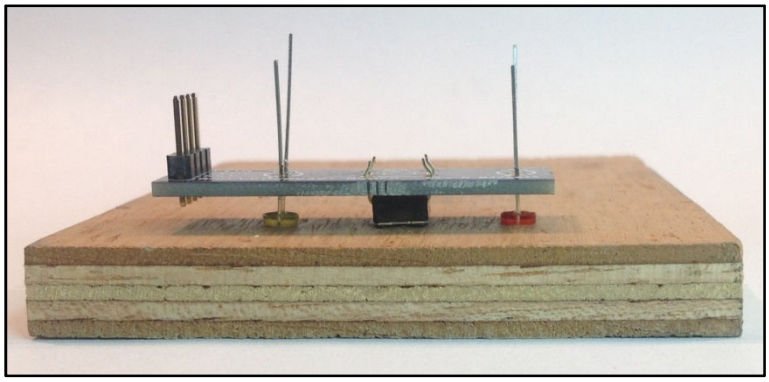


Figure 8 – Using a Jig to Align LED/Reset Board Components

- Finally, solder the 4-pin header to the board, with the header pins on the top (silkscreen) side of the board.
- If a reset switch is not being fitted, solder the header pins first, then mount the LEDs on the bottom of the board, ensuring that the LEDs are spaced sufficiently far off the PCB to provide clearance for the solder side of the header pins.

Enclosure

Two alternative enclosures are suggested for the Simulator Interface, the MB7 and MB4 Black ABS cases manufactured by BCL Distribution Ltd of Harrogate. These can be obtained from Maplin and ESR Components.

- The MB7 enclosure is 177mm x 120mm x 83mm overall, and is suitable for interfaces with up to eight sensor connectors.
- The larger MB4 enclosure is 216mm x 130mm x 85mm overall, and is suitable for interfaces with 10 or 12 sensor connectors.

Parts List

Table 6 – Simulator Interface Enclosure Parts List

Component	Notes
MB7 ABS Black Enclosure MB4 ABS Black Enclosure	ESR Components 400-580, Maplin KC89W ESR Components 400-565, Maplin LH23A
M3 x 12mm Nylon Hexagonal Threaded Spacer	Maplin QT80B
M3 x 6mm Nylon Pan Head PCB Fixing Screw	eBay

Construction

The following diagrams give suggested drilling layouts for each type of enclosure.

- In both cases the enclosures are used with the lid at the bottom, the sensor connectors, etc, mounted through the base, and the Simulator Interface PCB mounted on the inside of the lid. This allows easier access to the PCB and connections, for example for uploading firmware.
- Drilling large diameter holes with twist drills can result in bit grabbing and damage to the enclosure. Small diameter (13mm/16mm/18mm) hole saws are available at relatively low cost (e.g. on eBay), and these make the process of drilling the enclosure much easier and safer.
- If the optional reset switch is not being fitted, omit the central 4.5mm hole between the two diagnostic LEDs.
- Using a sharp chisel, carefully and gently remove the PCB ribs from the inside of the end of the enclosure through which the power/data connector will be mounted. Using too much force is likely to crack the case.

MB7 Enclosure (Up to 8 Sensors)

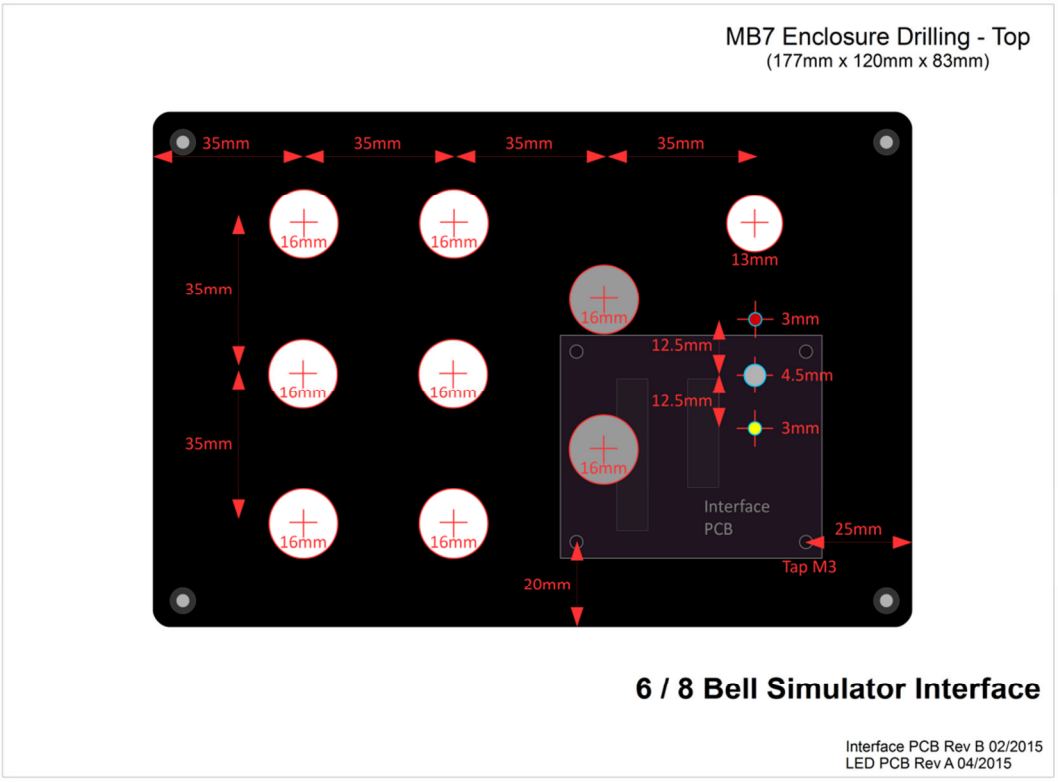


Figure 9 – MB7 Drilling Guide – Top

Figure 10 – MB7 Drilling Guide – End

MB4 Enclosure (10/12 Sensors)

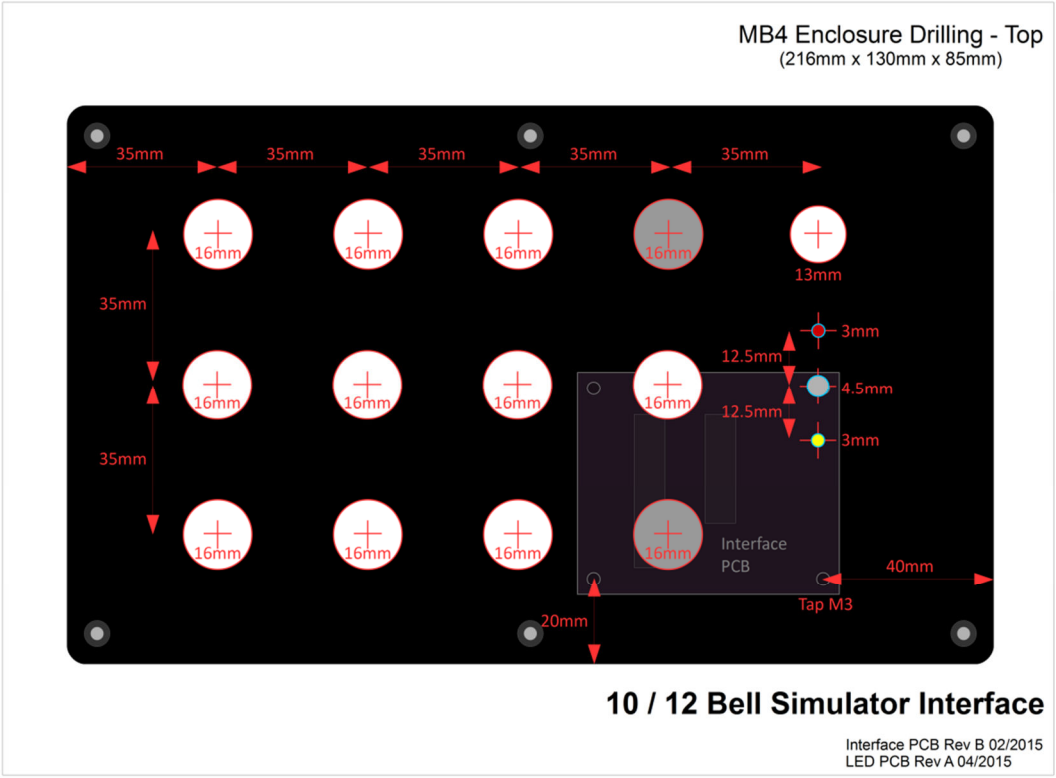


Figure 11 - MB4 Drilling Guide – Top

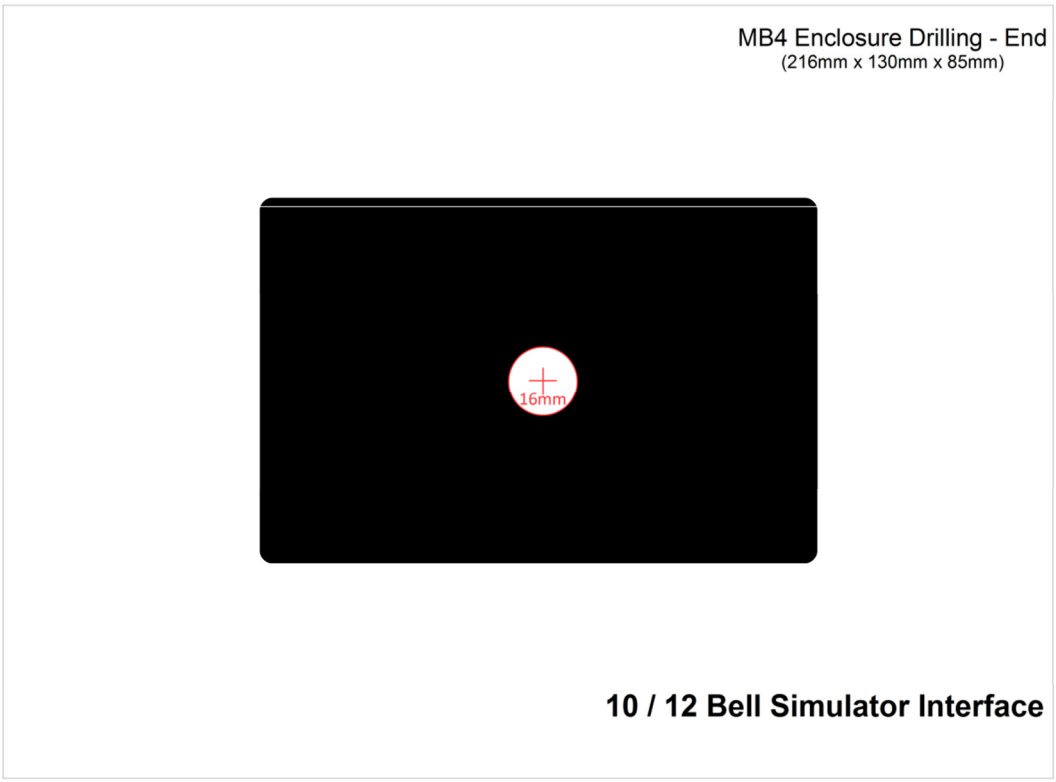


Figure 12 – MB4 Drilling Guide – End

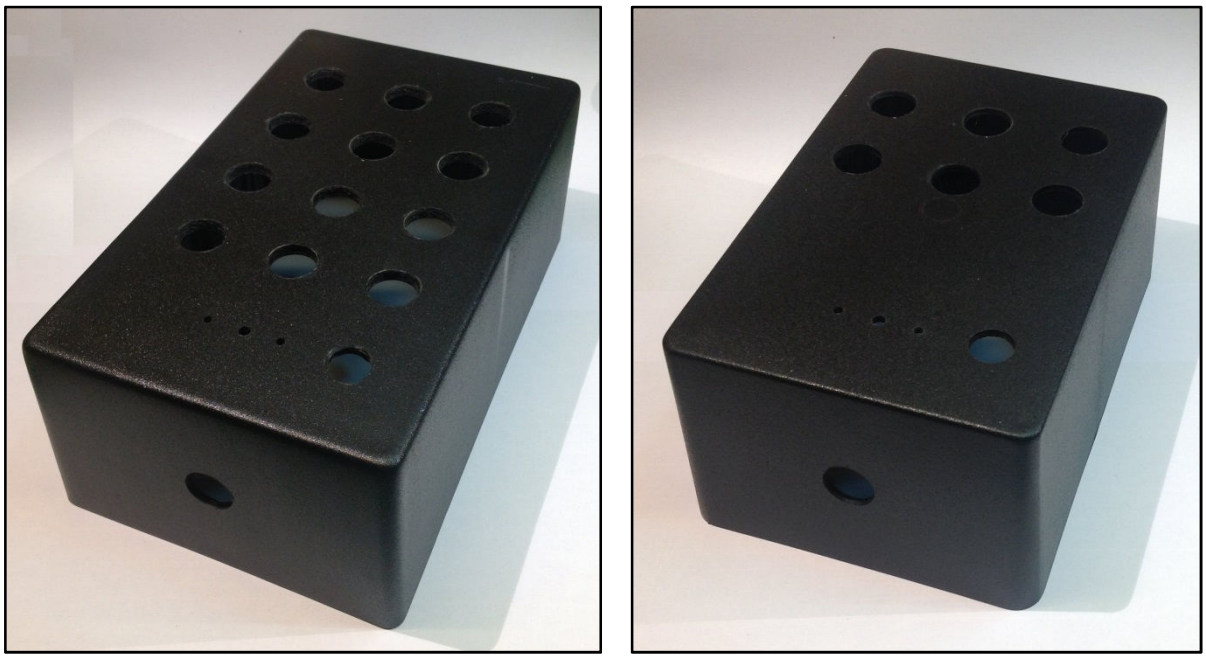


Figure 13 – Drilled MB4 & MB7 Enclosures

PCB Mounting

The Simulator Interface PCB is mounted on threaded nylon standoffs on the inside of the enclosure lid. Drill the mounting holes at 2.5mm, and then tap to M3. The standoffs can then be screwed into the tapped holes from the inside of the enclosure, and the excess thread trimmed off with a sharp knife or chisel.

The following photograph shows a 6-bell PCB mounted on the inside of the lid of a MB7 enclosure, and examples of the standoffs and mounting screws used.

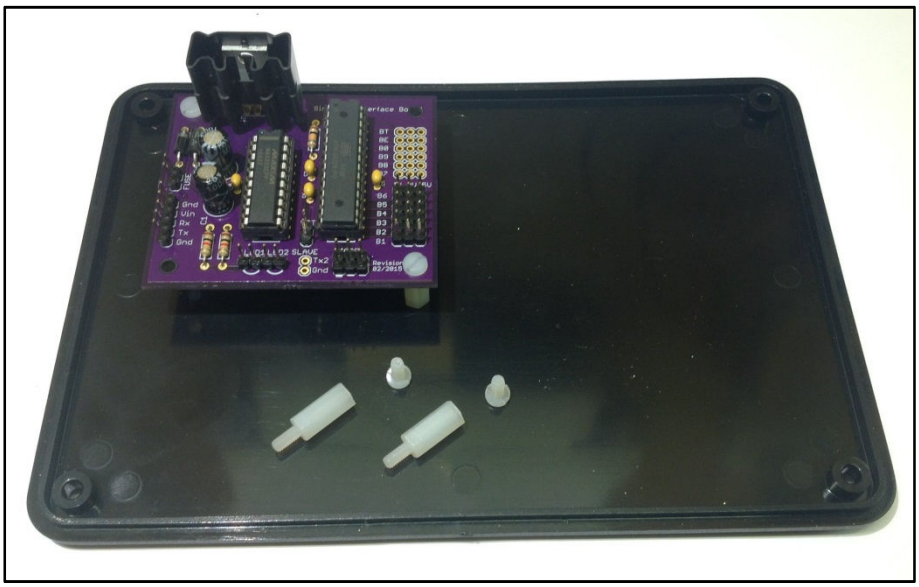


Figure 14 – Interface PCB Mounting

The LED/Reset Board is held in place inside the base of the enclosure with a small amount of hot melt glue. This is visible in one of the assembly photographs below.

Internal Cabling

A number of internal cables are required to interconnect the components of the Simulator Interface, and for the external connectors. This section describes the following cables:

- Sensor Connector Cables
- LED/Reset Board Cable
- Power/Data Cable
- Fuse Holder Cable

Parts List

Table 7 – Internal Cabling Parts List

Component	Notes
20mm Panel Mount Fuse Holder	Rapid 26-0191, Maplin DA59P
20mm 800mA Quick Blow Fuse	Maplin GJ89
GX16-4 Chassis Plug	Sensor Connectors as required (1 – 12)*
GX16-5 Chassis Plug	Power/Data Connector
Heat Shrink Sleeving 2.5mm Un-shrunk Size	
Ribbon Cable	
DuPont 0.1” Female Crimp Connectors	11, plus 3 – 36 as required*
DuPont 0.1” Female Connector Shells	1 x 5-pin, 1 x 4-pin, 2 x 2-pin, plus 1 to 12 x 3-pin as required*

(* Depending on the number of Sensors Connected)

While it is possible to make up the DuPont connector cables by hand, this is time consuming and not easy even if the proper crimp tool is available. An alternative approach is to buy ready-made lengths of ribbon cable with female DuPont connectors already crimped, and cut these down to make the required cables. These are readily available on eBay, and can save a lot of work.

Note that the core colours shows in this section are for clarity only, and have no other significance.

Sensor Connector Cables

The Sensor Connector Cables have a GX16-4 chassis plug on one end, and a 3-pin female DuPont connector on the other, wired as shown the following diagram.

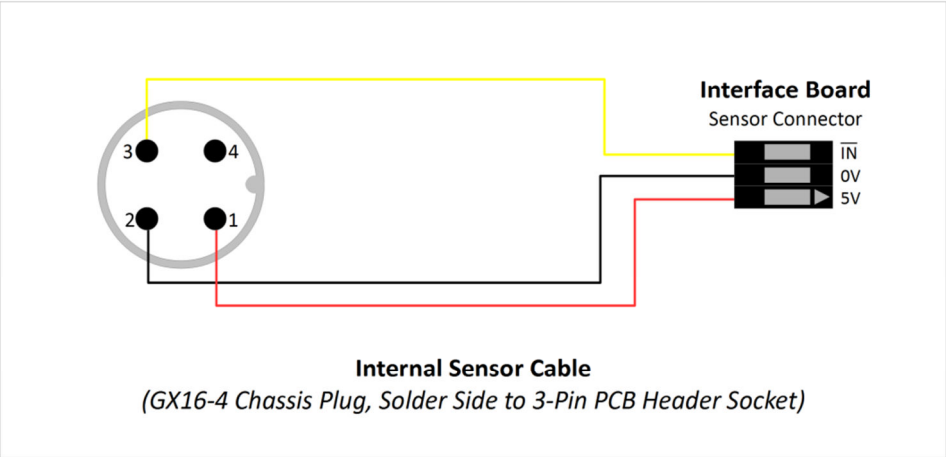


Figure 15 – Internal Sensor Cable Diagram

If Sensor Heads using the standard infra-red detectorin the suggested enclosure are also being constructed, suitable lengths of cable complete with 3-pin connectors can be made by cutting approximately 16cm of cable off the detector, leaving a short pigtail for connection to the corresponding GX16-4 connector on the sensor enclosure. Otherwise use a suitable length of ribbon cable.

The cable ends are soldered onto the GX16-4 chassis plugs following the wiring the diagram above. A short length of heatshrink sleeve, 2.5mm unshrunk diameter, is recommended on each core to protect the soldered connection and provide some strain relief. A completed set of Sensor Connector Cables is shown in the following photograph.

Figure 16 – Internal Sensor Connector Cables

If cables from the infra-red sensors are used, double check that the order of the cores in the 3-pin connector is correct. If it is not, the crimped connections can be removed from the plastic shell by carefully prising up the small black tab and sliding the connector out.

Where two detectors are mounted in a single Sensor Head, a suitable 4-core cable may be made up instead, or a link wire soldered between the two relevant GX16-4 connectors.

LED/Reset Board Cable

The LED/Reset Board Cable has a 4-pin female DuPont connector on the LED/Reset Board end. The Interface Board end has two female DuPont connectors; a 2-pin connector for the reset switch connection, and a 4-pin connector with only two pins populated for the LED connection. This is wired as shown the following diagram.

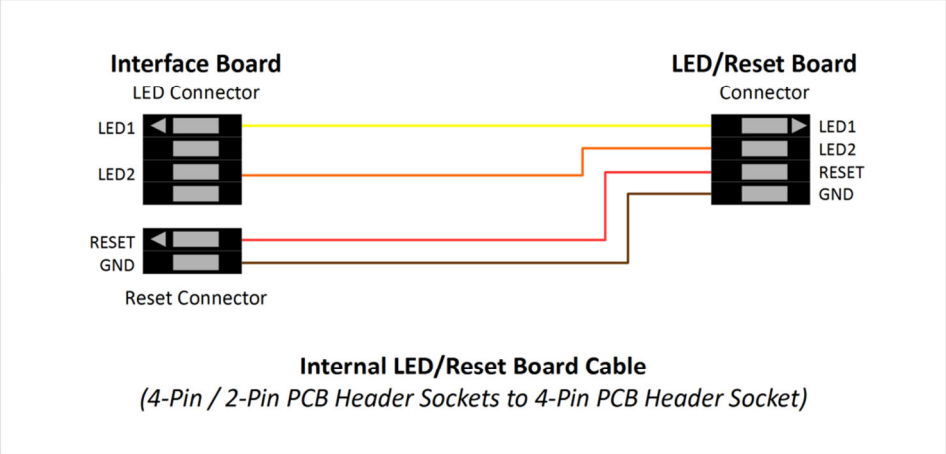


Figure 17 – Internal LED/Reset Board Cable Diagram

With the enclosures and drilling patterns suggested, the required length of the cable is approximately 12cm overall for the MB7 enclosure, and 20cm overall for the larger MB4 enclosure.

When connecting this cable to the Interface Board note that:

- The ground connection on the reset switch pins is the upper pin (nearest C6).
- The LED connector is oriented so that the left pin (nearest R3) of each is connected.

A completed LED/Reset Board Cable is shown in the following photograph.

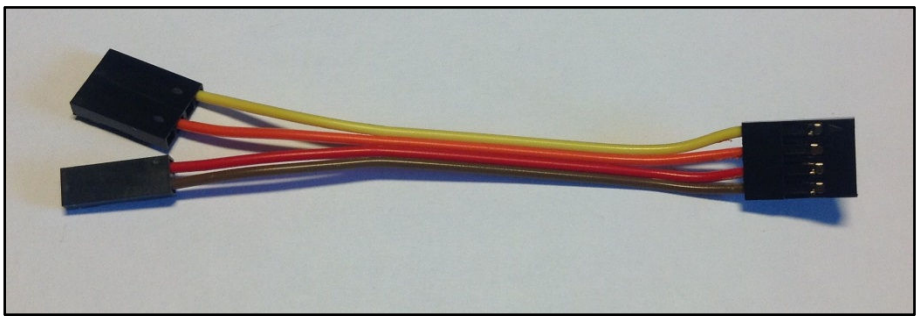


Figure 18 – LED/Reset Board Cable

Power/Data Connector Cable

The Power/Data Connector Cable has a GX16-5 chassis plug on one end, and a 5-pin female DuPont connector on the other, wired as shown the following diagram. Note that the connections from pins 2 & 3 to the Rx & Tx connections on the Interface Board are crossed over.

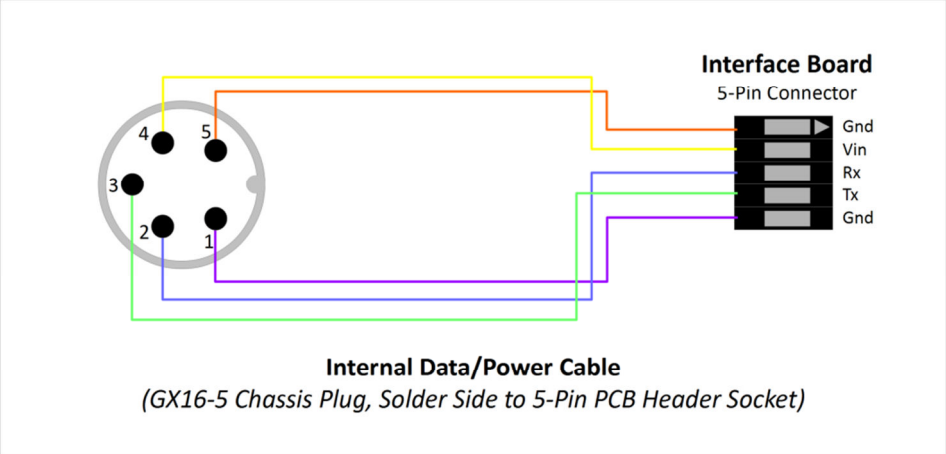


Figure 19 – Internal Power/Data Cable Diagram

A suitable length of ribbon cable can be used for this cable, the required length of the cable is approximately 12cm overall.

The cable ends are soldered onto the GX16-5 chassis plugs following the wiring the diagram above. A short length of heatshrink sleeve, 2.5mm unshrunk diameter, is recommended on each core to protect the soldered connection and provide some strain relief.

A completed Power/Data Connector Cables is shown in the following photograph. Note that this is a test cable and longer than required. Another example can be seen in one of the photographs in the Assembly section of this document.

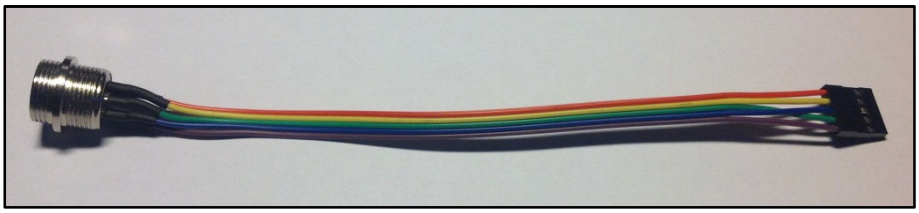


Figure 20 – Power/Data Connector Cable

Fuse Holder Cable

The Fuse Holder Cable has a 2-pin female DuPont connector on one end. The other end is soldered directly to the fuse holder. A short length of heatshrink sleeve, 2.5mm unshrunk diameter, is recommended on each core at the fuse holder end.

This cable can be seen in one of the photographs in the Assembly section of this document.

External Cabling

Two types of external cables are required to interconnect the Simulator Interface to the Sensor Heads and the Simulator PC. This section describes the following cables:

- Sensor Head Cables
- Power/Data Cable

Parts List

Table 8 – External Cabling Parts List

Reference	Component	Notes
Sensor Cables	3-Core Signal Cable, Def Stan 61/12 Part 4 Code 7-2-3a, or 4-Core Signal Cable 7-2-4a	Quantity and length as required (1 – 12) Rapid Electronics 02-0261 Rapid Electronics 02-0262
Sensor Cable Connectors	GX16-4 Line Socket	Quantity as required (2 – 24)
Cable Markers	Heatshrink Cable Numbers	Optional. Quantity and markings as required.
Power/Data Cable	6-Core Signal Cable, Def Stan 61/12 Part 4 Code 7-2-6a	Length as required. Rapid Electronics 02-0263 (one core is unused).
Power/Data Connector	GX16-5 Line Socket	
Serial Connector	9-Pin Female DB9F Connector	
Serial Connector	9-Pin D Shell	
DC Power Connector	2.5mm x 5.5mm Power Connector	Maplin JK12N
Heat Shrink Sleeving	Assorted Sizes	

Sensor Head Cables

One Sensor Head Cable is required for each Sensor Head connected to the Simulator Interface. The wiring of each Sensor Head Cable is shown in the following diagram.

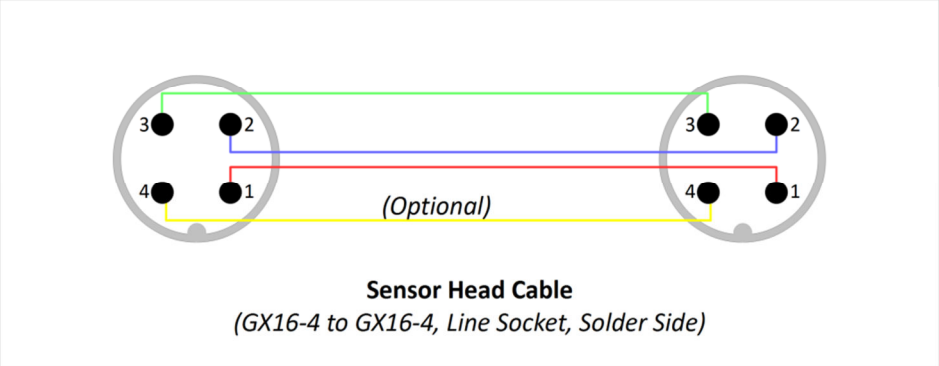


Figure 21 – Sensor Head Cable Wiring

The cable used is Def Stan 61/12 Part 4 Code 7-2-3a multicore cable. Individual cable lengths up to 11m have been tested successfully so far. For a double detector Sensor Head, 4-core cable such as Def Stan 61/12 Part 4 Code 7-2-4-a may be used.

The core colours used in the Sensor Head Cables are shown in the following table.

Table 9 – Sensor Head Cable Wiring Colours

Pin	Colour
1	Red
2	Blue
3	Green
4 (Optional)	Yellow

A completed Sensor Head cable is shown in the following photograph.

Figure 22 – Sensor Head Cable

Power/Data Cable

A single Power/Data Cable is required to connect the Simulator Interface to the Simulator PC, and to supply power to the Interface from a plug-in power supply. The wiring of the Simulator Data Cable is shown in the following diagram.

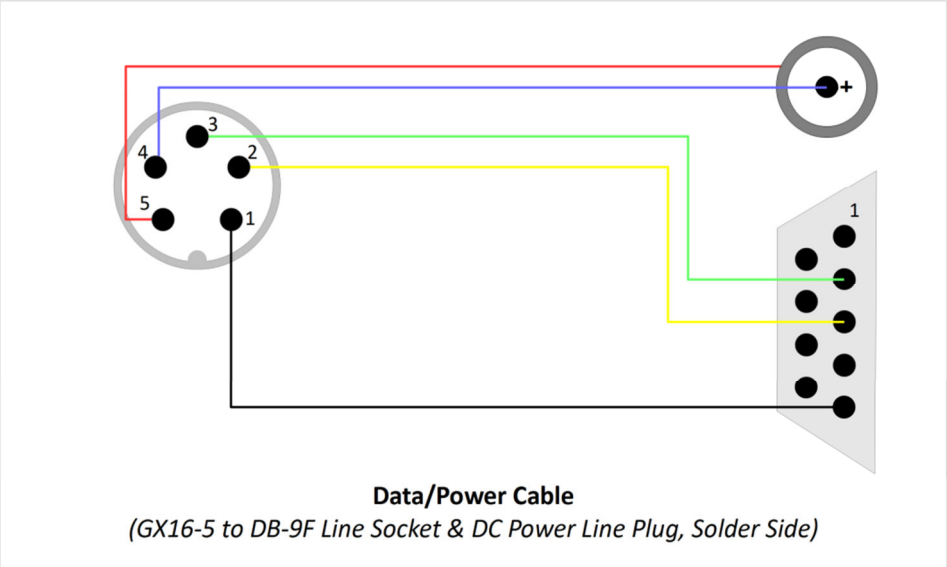


Figure 23 – Simulator Data Cable Wiring

The cable used is Def Stan 61/12 Part 4 Code 7-2-6a multicore cable, with one core unused. Cable lengths up to 25m have tested successfully.

The DC power connector is 2.5mm x 5.5mm, centre pin positive. The power supply cores are soldered to a pigtail of twin core cable within the D-connector cover shell, and the joints covered in heatshrink sleeving. The pigtail is then brought out through the rear cable entry and soldered to the power connector.

The core colours used in the Simulator Power/Data Cable are shown in the following table.

Table 10 – Simulator Data Cable Wiring Colours

Pin	Colour
1	Black
2	Yellow
3	Green
4	Blue
5	Red
N/C	White

A completed test Power/Data Cable is shown in the following photograph.

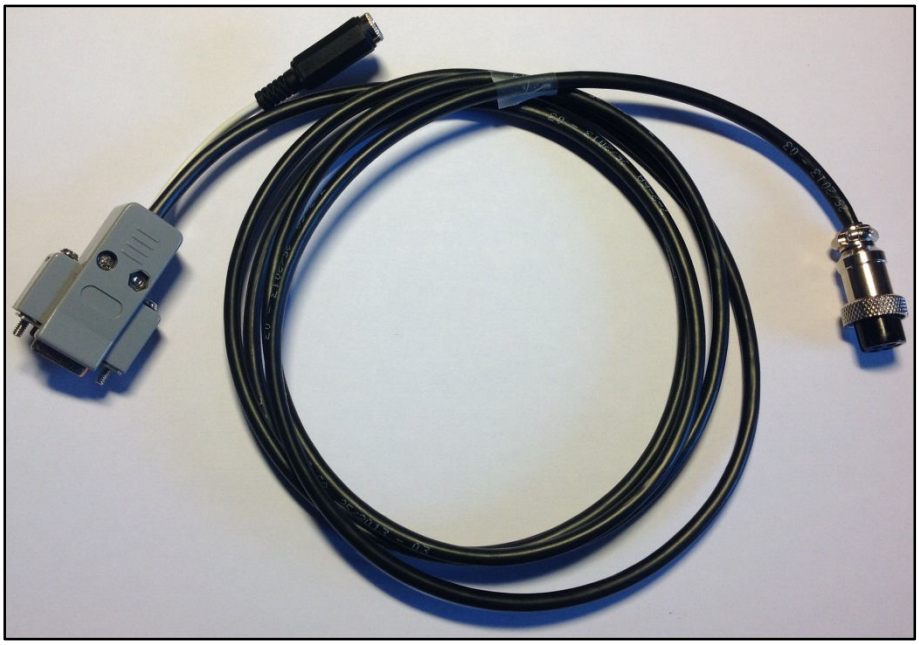


Figure 24 – Power/Data Cable

Power Supply

A plug-in power supply is required to supply power to the Simulator Interface.

- A regulated DC power supply rated at 1A with multiple selectable output voltages is recommended, for example Maplin part number L82BF or equivalent.
- The output connector required is 2.5mm x 5.5mm, centre pin positive.
- The output voltage of the power supply should be adjusted so that the supply voltage at the Simulator Interface is at least 7.5 volts, measured at the cathode (striped) end of D1, with all Sensor Heads connected.
- The supply voltage may be higher than that required to maintain 7.5 volts at the Interface, but this may result in a larger heatsink being required for the voltage regulator IC1.
- As a guideline, a supply voltage of 9V has been found sufficient to maintain the required voltage on the Liverpool Cathedral installation, with all 12 Sensor Heads connected and a 24m Power/Data cable.

Firmware Upload

The firmware for the Simulator Interface Board is released under the GNU General Public Licence (GPL), Version 3, and the source code and other supporting files can be downloaded from GitHub.

- https://github.com/Simulators/simulator/tree/master/firmware/simulatorinterface

The Simulator Interface firmware is held in non-volatile flash memory on the ATmega328P microcontroller. It should only be necessary to re-upload the software in the event that the microcontroller is replaced, the flash memory has become corrupted, or the Simulator Interface firmware requires updating.

The firmware code needs to be uploaded to the microcontroller on the Simulator Interface PCB. Although the software development environment (described in the Software Manual) is based on the Arduino platform, the Simulator Interface does not use the Arduino bootloader, and it is not possible to upload the firmware over the interface's RS-232 serial port. Firmware is uploaded using a hardware programmer via the ICSP header pins provided on the interface PCB.

There are two main options for the hardware programmer:

- A dedicated hardware ISP programmer such as the *AVR ISP*[29].
- An Arduino add-on board or shield such as the *Arduino ISP*[30] or the BoardStuff *UNO Multi Programming Shield*[31].
- An Arduino board (with one additional component) used as an ISP programmer.

The last of these requires no special hardware, and is the approach described in this document. There are also many tutorials online, including on the Arduino website[32].

[29] http://www.atmel.com/tools/avrispmkii.aspx
[30] http://www.arduino.cc/en/Main/ArduinoISP
[31] http://www.boardstuff.co.uk/
[32] http://www.arduino.cc/en/Tutorial/ArduinoISP

Preparing the Environment

Perform the following steps to prepare the PC software environment for compiling and uploading the Simulator Interface firmware:

- Download and install the latest Arduino IDE package[33]. At the time of writing this was version 1.6.3.
- Start the IDE, and locate the Arduino sketchbook directory by selecting *File | Preferences*.

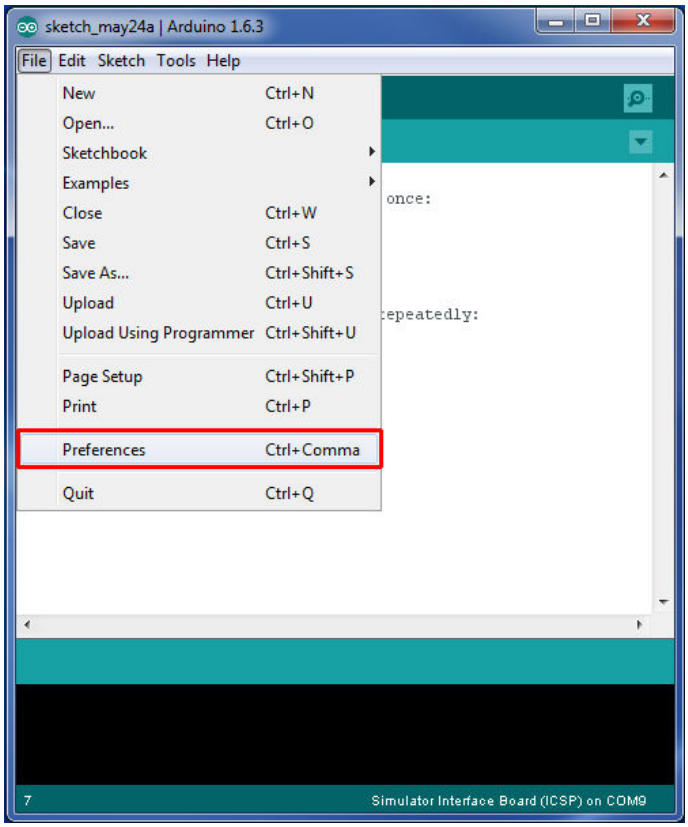


Figure 25 – Arduino IDE Preferences Menu

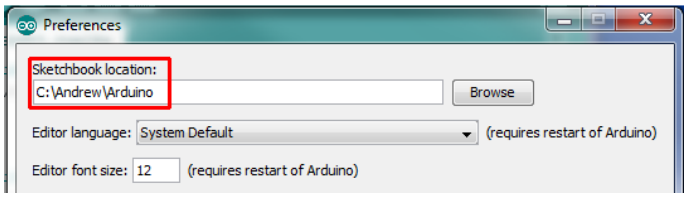


Figure 26 – Arduino IDE Sketchbook Location

- Under the Arduino sketchbook directory, create the following directory path: `hardware\simulators\avr\bootloaders`

[33] http://www.arduino.cc/en/Main/Software

- In the new `avr` directory, create a text file called `boards.txt`. This file defines the parameters for uploading to the Simulator Interface hardware. This includes the necessary fuse settings to enable the internal 8MHz clock. The file can be found in the GitHub repository, and the current version looks like this:

```
# Simulator Interface Board - HW Rev B
simulator.name=Simulator Interface Board (ICSP)
# Upload
simulator.upload.using=arduino:arduinoisp
simulator.upload.maximum_size=32768
simulator.upload.tool=arduino:avrdude
# Bootload
simulator.bootloader.low_fuses=0xE2
simulator.bootloader.high_fuses=0xD7
simulator.bootloader.extended_fuses=0x07
simulator.bootloader.tool=arduino:avrdude
simulator.bootloader.file=optiboot_atmega328.hex
simulator.bootloader.unlock_bits=0x3F
simulator.bootloader.lock_bits=0x3F
# Build
simulator.build.mcu=atmega328p
// Alternative mcu
//simulator.build.mcu=atmega328
simulator.build.f_cpu=8000000L
simulator.build.core=arduino:arduino
simulator.build.variant=arduino:standard
```

- **Important note:** If an ATMega328-PU microcontroller is being used instead of the ATMega328P-PU, change the `simulator.build.mcu` line to reflect the alternative microcontroller. If this is not done, programming will fail with an *Invalid Signature* error.
- Copy the file `optiboot_atmega328.hex` from the main Arduino installation directory into the new `bootloaders` directory. Although the Simulator Interface does not use the Arduino bootloader, fuse settings are only applied during the *burn bootloader* programming phase. The uploaded bootloader will be overwritten when the main firmware is uploaded to the microcontroller.
- Re-start the Arduino IDE.

The environment is now ready to set up the programmer.

Preparing the Programmer

The programmer is an unmodified Arduino Uno board running a sketch which allows it to operate as an ISP programmer.

This requires an Arduino Uno board, and a Type A to Type B USB cable (sometimes known as a printer cable).

Figure 27 – Arduino USB Cable

The Arduino website has instructions[34] on connecting the Arduino board to a computer, installing drivers and setting up the IDE.

Perform the following steps to prepare the programmer Arduino Uno board:

- Connect the “B” end of the USB cable to the Arduino Uno board to be used as the programmer. From now on this board is referred to simply as “the programmer”.
- Connect the “A” end of the USB cable to the computer.
- Follow the instructions on the Arduino site to install drivers (if necessary), and select the correct port and board type for the programmer in the IDE.

[34] http://arduino.cc/en/guide/windows

- Open the *ArduinoISP* software sketch (supplied as part of the default IDE installation) in the Arduino IDE by selecting it from the *File | Examples* menu.

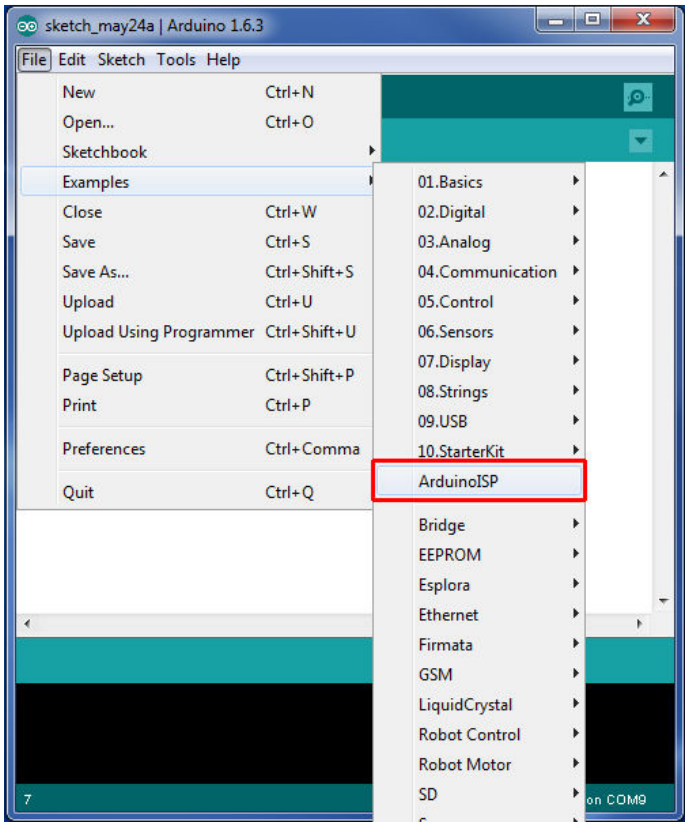


Figure 28 – Arduino IDE ISP Sketch Loading

- On the *Tools* menu, ensure the correct board type for the programmer is selected ("*Arduino Uno*", not "*Simulator Interface (ICSP)*") and port. Correct these if necessary.

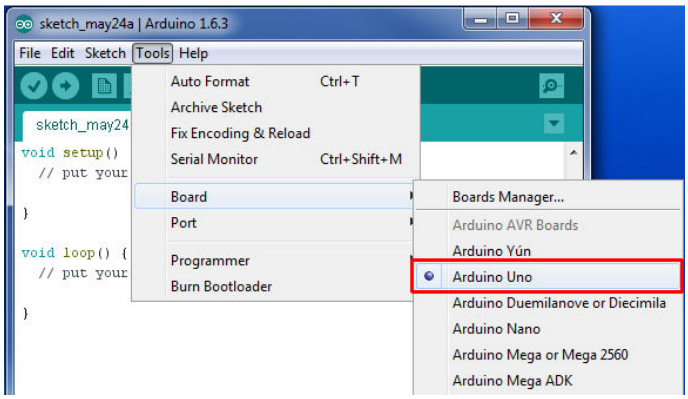


Figure 29 – Arduino Programmer Board Selection

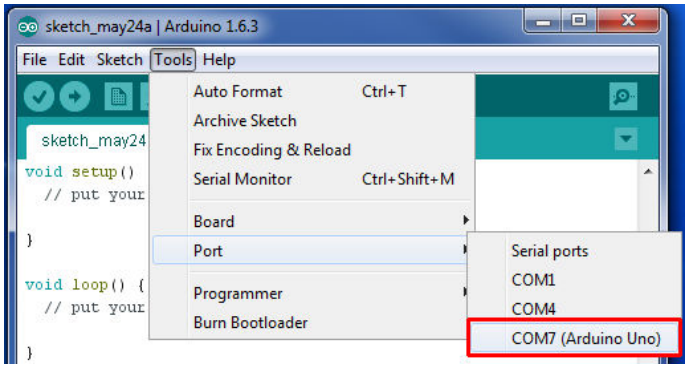


Figure 30 – Arduino Programmer Port Selection

- Click the upload (arrow) button on the IDE toolbar. The *ArduinoISP* code will be compiled and uploaded to the programmer. Verify that the upload completed successfully by looking for the "*Done uploading*" message.

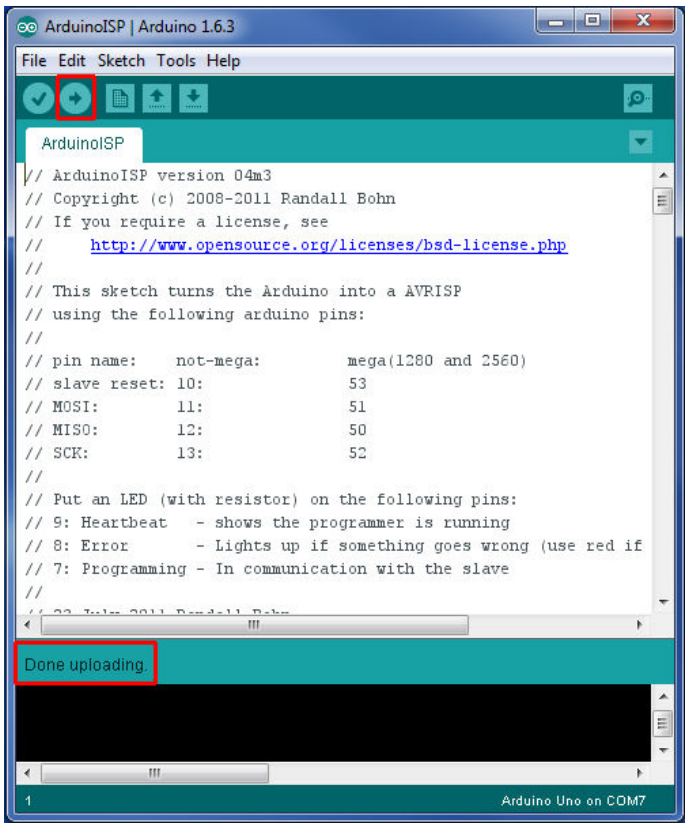


Figure 31 – Arduino IDE ISP Upload

- A failed upload will be indicated by error messages in the status area at the bottom of the IDE window.
- Disconnect the USB cable from the programmer.

- Connect a 10µF 25V electrolytic capacitor between the Reset and Ground pins of the programmer, negative side to Ground. This prevents the IDE from resetting the programmer and overwriting the *ArduinoISP* software.

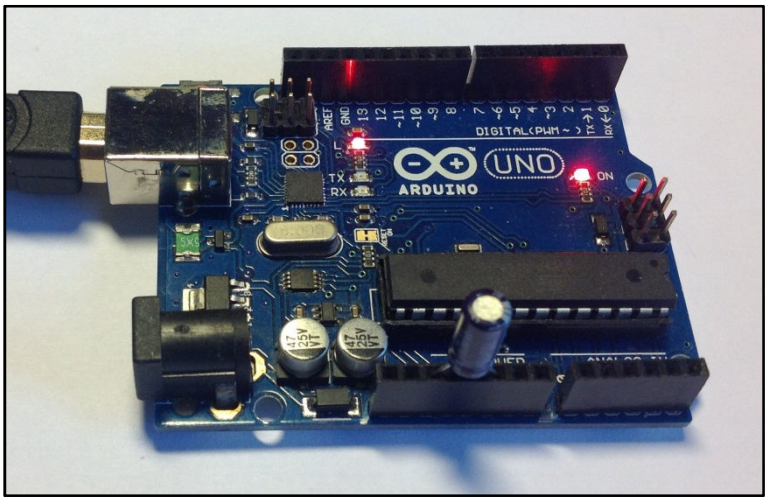


Figure 32 – Programmer with Capacitor

- Reconnect the USB cable to the programmer.

The programmer is now ready for use.

Setting the Fuses

Perform the following steps to set the microcontroller fuses. The fuses and their values are explained in a previous section of this manual.

- Disconnect the USB cable from the programmer.
- Connect the ICSP pins on the Simulator Interface to the ICSP pins on the programmer with jumper wires as shown in the following diagram.
- Pin 1 on the Simulator Interface PCB is bottom left, identified by a white stripe.
- Pin 1 on the programmer is top left. Note that pin 5 on the Simulator Interface PCB is connected to pin 10 on the programmer.

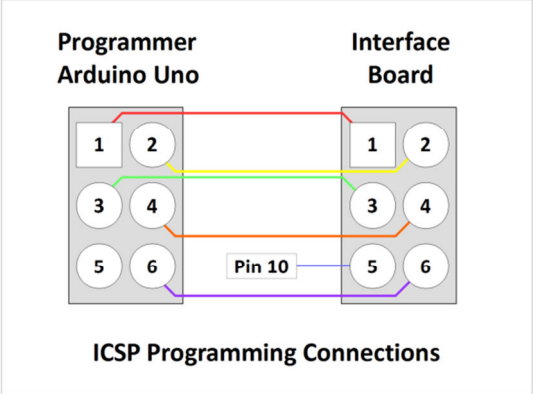


Figure 33 – Programmer Connections

- The following photograph shows the programmer connected to an interface board, including the connection to pin 10 of the programmer (orange wire), not to the ICSP pin. (Note also that the Simulator Interface Board is fitted with the alternative Tracopower voltage regulator.)

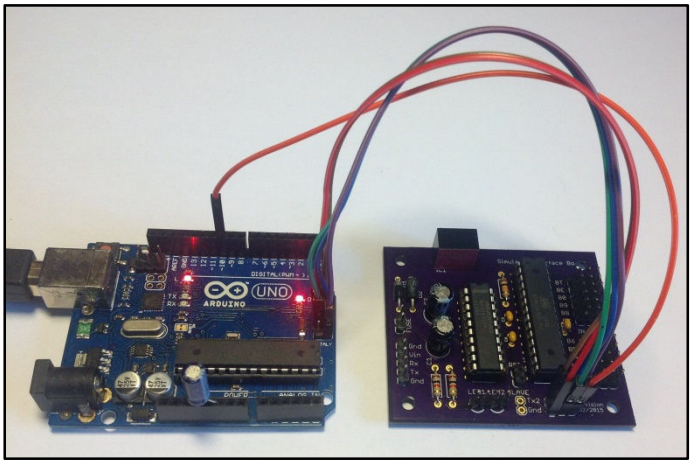


Figure 34 – Programmer Connected to Interface Board

- Reconnect the USB cable to the programmer.

- On the *Tools | Board* menu, ensure the correct target board type to be programmed has been selected, in this case "*Simulator Interface (ICSP)*".

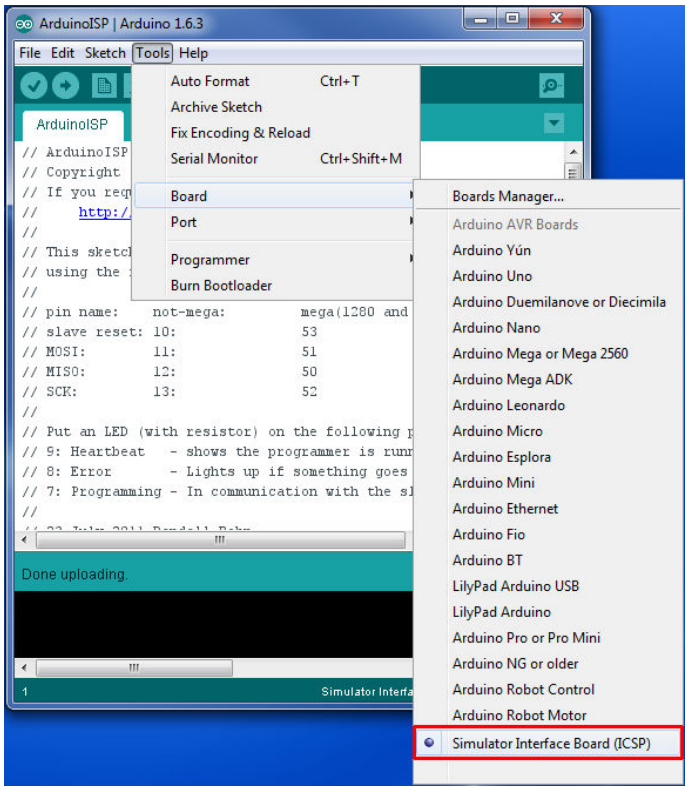


Figure 35 – Arduino IDE Target Board Selection

- On the *Tools | Programmer* menu, select *Arduino as ISP* as the programmer type.

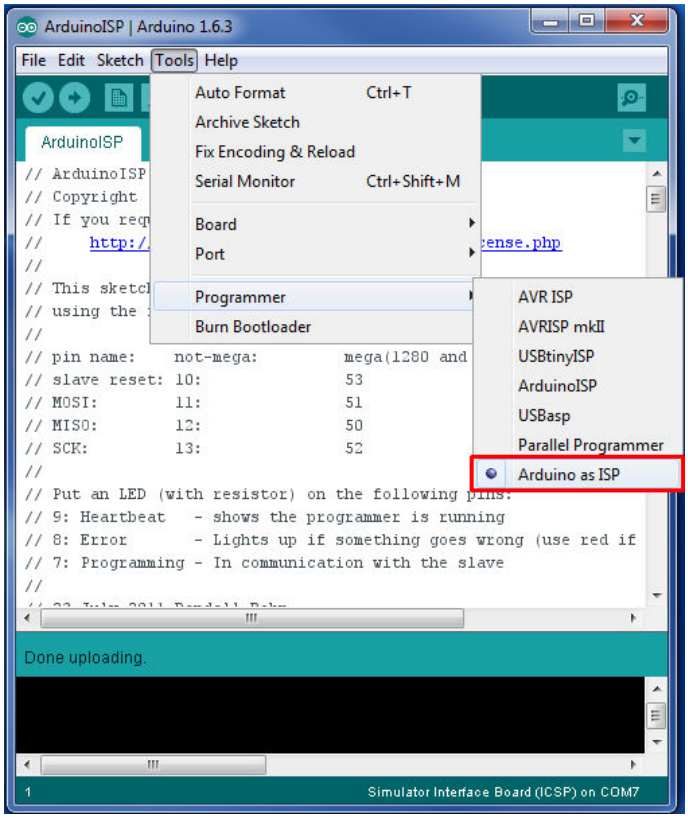


Figure 36 – Arduino IDE Target Board Selection

- On the *Tools* menu, select *Burn Bootloader*. The bootloader will be written to the Simulator Interface board, and the microcontroller fuses will be set. Verify that the burn process completed successfully by looking for the "*Done burning bootloader*" message.

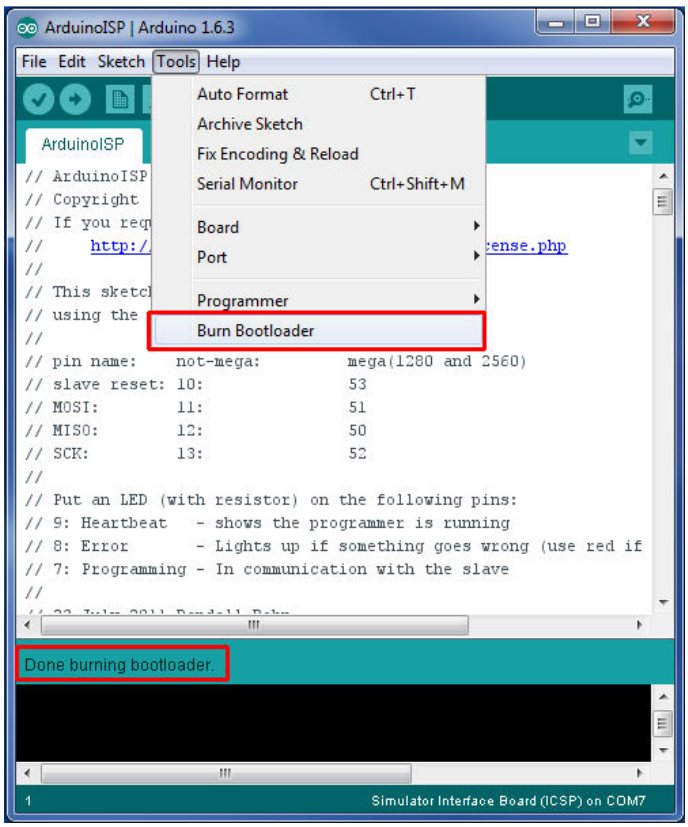


Figure 37 – Arduino IDE Burn Bootloader

- **Important note:** If a microcontroller previously used in an Arduino board is to be re-used on the Simulator Interface board, carry out the steps above to set the fuses before removing the microcontroller from the donor Arduino. Brand new ATmega328P-PU microcontrollers should be configured to use the 8MHz internal clock by default, but ones previously used on an Arduino will be configured to require an external crystal clock. Once you have set the fuses, move the microcontroller from the donor Arduino to the Simulator Interface Board.
- Note that if new firmware is being uploaded to an existing Simulator Interface Board, there should be no need to go through the steps to set the fuses every time, unless a change in fuse values is required by the new firmware.

The microcontroller is now ready for firmware upload.

Firmware Upload

Perform the following steps to upload the Simulator Interface firmware to the board.

- Connect the Simulator Interface Board to the programmer as described in the previous section.
- Download and install the MemoryFree[35] and VTSerial[36] libraries. For convenience these libraries are can also be found in the GitHub repository with the Simulator Interface firmware. Note that the libraries can be installed straight from the zipfiles by selecting *Add .ZIP Library* from the *Sketch | Include Library* menu.

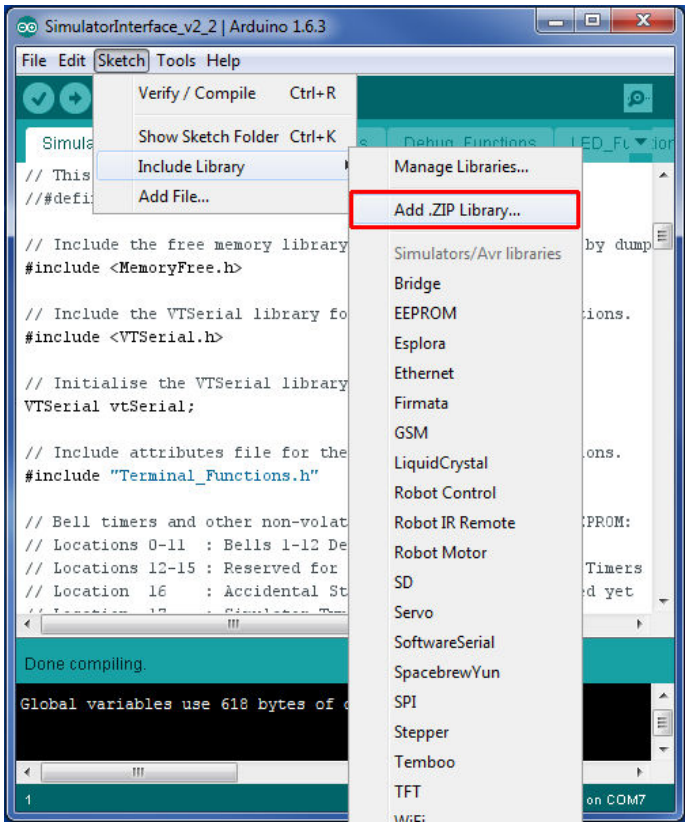


Figure 38 – Arduino IDE Add Library

- Download the Simulator Interface firmware and load it into the Arduino IDE by double clicking the name of the main file, e.g. *SimulatorInterface_v2_2.ino*.
- On the *Tools | Board* menu, as above ensure that the correct board type to be programmed has been selected, in this case "*Simulator Interface (ICSP)*".

[35] https://github.com/maniacbug/MemoryFree
[36] http://www.hobbytronics.co.uk/tutorials-code/arduino-tutorials/arduino-vtserial-library

- On the *Tools | Programmer* menu, as above select *Arduino as ISP* as the programmer type.
- Click the upload (arrow) button on the IDE toolbar. The Simulator Interface firmware will be compiled and uploaded to the interface board. Verify that the upload completed successfully by looking for the "*Done uploading*" message.

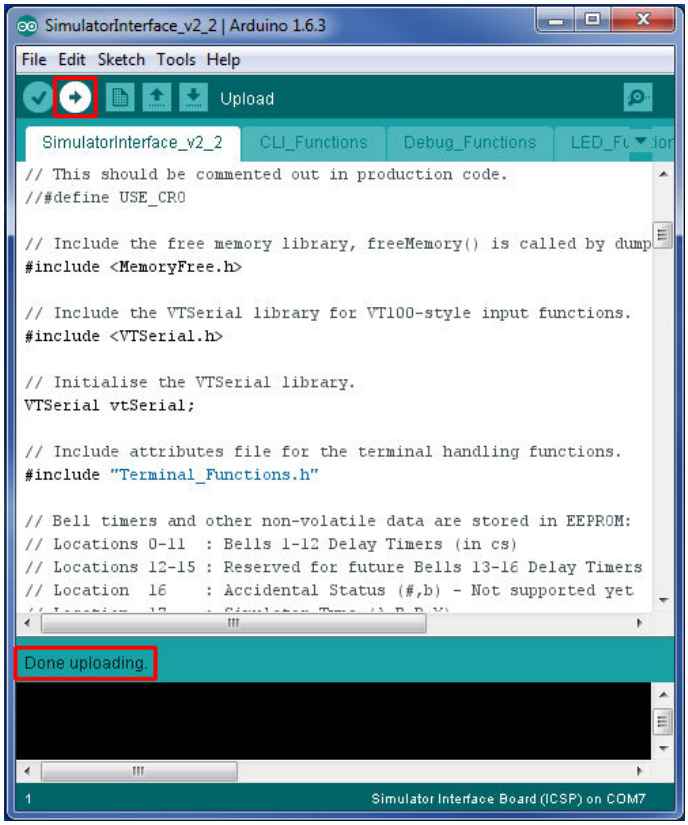


Figure 39 – Arduino IDE Firmware Upload

- A failed upload will be indicated by error messages in the status area at the bottom of the IDE window.
- When the upload has completed the Simulator Interface board will be reset, and on restarting the yellow diagnostic LED will flash according to the firmware version, for example two long and two short flashes indicates firmware version 2.2.
- Disconnect the USB cable from the programmer.
- Disconnect the programmer from the Simulator Interface Board.
- Note that when uploading new firmware to an existing Simulator Interface Board, the Sensor Head Cables and the Power/Data Cable must be disconnected from the Simulator Interface. There is no need to disconnect all the internal cabling from the PCB.

The Simulator Interface board now has the firmware installed, and is ready for final assembly.

Interface Assembly

Final assembly of the components is as follows.

- Carefully press fit the LEDs into the holes in the base of the enclosure. Ensure that the reset switch (if fitted) is free to move, and then secure the LED/Reset Board with a small quantity of hot-melt glue.
- Fit all the Sensor Connector Cable assemblies to the base of the enclosure and tighten the securing nuts. A plumber's box spanner is useful for this, and on the larger MB4 enclosure can be used to tighten the nut on the Power/Data Connector Cable assembly as well.
- Fit the fuse holder to the enclosure, taking care not to overtighten the plastic securing nut.
- Fit the LED/Reset Board Cable to the LED/Reset Board.
- Connect all the cables to their respective pins on the Simulator Interface Board, and carefully fit the lid to the enclosure using the screws supplied.
- Insert an 800mA quick blow fuse into the fuse holder, and screw the top in place.

The following photograph shows the inside of a 6-bell Simulator Interface during final assembly. Note that the PCB ribs have been removed from the inside of the right-hand end of the box, to allow the Power/Data Connector to be fitted.

Figure 40 – Internal View During Assembly

Two examples of completed Simulator Interfaces are shown in the following photographs.

Figure 41 – Completed 12 & 6 Bell Simulator Interfaces

Installation

Simulator Interface

The Simulator Interface is located in the belfry, in a location convenient for routing the Sensor Head and Power/Data Cables.

- The use of a secondary enclosure is recommended, especially where the belfry is not fully weather tight.
- A 9 litre "Really Useful" storage box[37] can be used as a secondary enclosure. A section of the lip of the box can be removed, with cable entry arranged so as to prevent water ingress.

The following picture shows the Simulator Interface at Liverpool Cathedral in its secondary enclosure.

Figure 42 – Installed Simulator Interface

Simulator Power/Data Cable

The Simulator Power/Data Cable is routed from the Simulator Interface down to the Simulator PC

- The cable should be secured so as to prevent the weight of the cable pulling on the connectors.
- The minimum diameter of any holes along the cable route is approximately 20mm, to pass the locking ring on the GX16-5 connector which is approximately 18mm in diameter.
- Alternatively, if it is practicable to solder connectors in-situ, a minimum hole diameter of approximately 6mm is adequate for the suggested cable type.

[37] http://www.reallyusefulproducts.co.uk