# The Liverpool Ringing Simulator

## One Bell Simulator Interface

## Hardware Manual

Author: Andrew Instone-Cowie

Date: 25 September 2015

Version: 1.2

# Contents

## Index of Figures

# Index of Tables

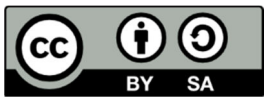
## Document History

| Version | Author | Date | Changes |
|---|---|---|---|
| 1.0 | A J Instone-Cowie | 05/08/2015 | First Issue (Main PCB Rev A, Firmware 1.1) |
| 1.1 | A J Instone-Cowie | 15/08/2015 | Added details of Revision C magneto-resistive sensor PCB as alternative to infra-red sensor |
| 1.2 | A J Instone-Cowie | 25/09/2015 | Tested against Abel 3.9.1. |
| | | | |
| | | | |
| | | | |



*Cover photograph: A completed One 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 a sensor, attached to a real tower bell or teaching dumb bell, to be connected to a computer Simulator Software 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.

This design is derived from the Liverpool Ringing Simulator Project Simulator Interface, a separate Hardware Manual for which is available, and which contains more details of the background to the project and the design of the interfaces.

## 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. More details of the background to and objectives of the project can be found in the accompanying Software Manual, or in the main Simulator Interface Hardware Manual.

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 a single bell 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 Single Bell Interface (SBI), complete with sensor head 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.

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

# One Bell Simulator Interface Design

## Overview

### Typical Simulator Installation

The following diagram illustrates the general arrangement of a Simulator installation using the One Bell Simulator Interface.

An integrated Sensor Head and Simulator Interface (the *One Bell Simulator Interface*) is located in the belfry. A single data cable transmits the signal from the One Bell 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 belfry to power the One Bell Simulator Interface.

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

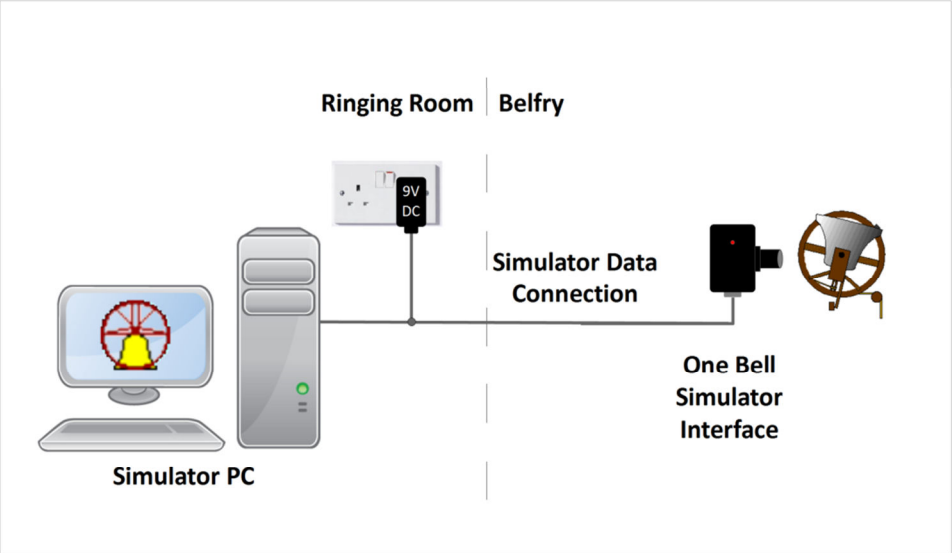



Figure 1 – Simulator General Arrangement

This manual describes the design and construction of the One Bell Simulator Interface hardware component of the Simulator. An accompanying Software Manual describes the Simulator Interface firmware, and the configuration of popular Simulator Software Packages.

A separate Hardware Manual is available which describes the design and construction of a Simulator Interface which supports the installation of sensors on up to 12 bells.

## 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. |
| One Bell Simulator Interface | A single Sensor Head and a Simulator Interface integrated into a single unit, for use with a single tower bell or dumb bell. |
| 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

A fuller discussion of these topics can be found in the main Simulator Interface Hardware Manual.

### 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.

- This design uses an optical sensor, using the low cost commercially available modulating infra-red detector adopted as the project standard.
- An alternative construction using a PCB developed for the magneto-resistive Sensor Head is also documented

### 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 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 before the Simulator triggers the simulated sound of the bell. Work by John Norris[12] 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 One Bell Simulator Interface adopts the single trigger approach, on the grounds of simplicity of installation and compatibility with other similar systems.

## One Bell Simulator Interface Hardware

### Overview

The One Bell 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 Head.
- A digital microcontroller, which detects incoming signals from the Sensor Head, applies a configurable delay, and then sends a signal 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.

[11] http://www.ringing.demon.co.uk/ringleader/ringleader.htm
[12] http://www.jrnorris.co.uk/strike.html

All the components of these functional blocks are mounted on a single custom PCB, together with diagnostic LED, and all components are mounted in a robust enclosure with a standardised connector for the Simulator Power/Data Connection.

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.

### Hardware Options

In common with the main Simulator Interface, the use of the same Atmel AVR microcontroller family has been continued for the One Bell Simulator Interface, allowing the same tool chain to be used for code development and deployment, and re-use of much of the firmware code.

### Microcontroller Hardware

The One Bell Simulator Interface uses the Atmel ATtiny85 microcontroller.

- The ATtiny85 is an 8-pin device with a much smaller footprint than the ATmega328P used in the main Simulator Interface. This allows the overall interface PCB size to be reduced, so that the PCB can be accommodated in the same enclosure adopted as the project standard for Sensor Heads.
- The One Bell Simulator Interface makes use of the ATtiny85's internal 8MHz oscillator. Although this is less accurate than an external oscillator, the internal oscillator is still adequate for the purposes of the interface, and this reduces the overall component count and complexity of the interface board by omitting a crystal and load capacitors.
- Although the One Bell Simulator Interface uses the Arduino development tool chain, it does not use the Arduino boot loader program. The ATtiny85 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.)
- Unlike the ATmega328P, the ATtiny85 does not have a hardware serial UART. The serial port is therefore emulated in the One Bell Simulator Interface firmware, using the standard Arduino *SoftwareSerial* library.

### Serial Line Driver

The Simulator Interface uses the Maxim MAX233 RS-232 serial line driver[13]. 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

A single diagnostic LED is included to provide visual status information on the activity of the One Bell Simulator Interface, because of the difficulty of using the serial interface for this purpose. This is optional, but recommended.

### Transient Voltage Suppression

The One Bell Simulator Interface 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 main Simulator Interface and Sensor Head designs).

[13] http://www.maximintegrated.com/en/products/interface/transceivers/MAX233.html

### Microcontroller I/O Pins

The following table shows the ATtiny85 pin assignments for the One Bell Simulator Interface:

**Table 2 – Microcontroller I/O Pin Assignments**

| Arduino I/O Pin[14] | ATtiny85 Physical Pin | Function |
|---|---|---|
| 0 | 5 | Red Diagnostic LED |
| 1 | 6 | (Spare) |
| 2 | 7 | Bell Sensor |
| 3 | 2 | Serial Port Transmit |
| 4 | 3 | Serial Port Receive |

### Microcontroller Fuse Settings

The ATtiny85 microcontroller has a number of configuration registers known as fuses. These are used to control the behaviour of the microcontroller[15]. 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 One Bell Simulator Interface are defined in the following table.

**Table 3 – ATtiny85 Fuse Settings**

| Fuse | Value | Notes |
|---|---|---|
| Low | 0xE2 | 8MHz internal oscillator, maximum clock startup delay. |
| High | 0XD7 | Enable serial programming, preserve EEPROM contents. |

For more details of these fuse settings and their functions, refer to the ATtiny85 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 One Bell Simulator Interface over the Power/Data Cable at a higher voltage than required, to overcome the effects of losses due the resistance of the cable. Incoming power is fed to a conventional linear 5V regulator on the One Bell Simulator Interface PCB. A small TO-92 case voltage regulator is used to reduce the overall PCB footprint.

### Typical & Peak Current Requirements

The One Bell Simulator Interface main board and sensor all operate at 5V DC. The typical total power requirement with a standard infra-red optical sensor is approximately 34mA, with the sensor not triggered. However the sensor draws more current when in the triggered state, mainly because the infra-red detector used includes a small internal status LED. The peak power requirement with the sensor in the triggered state is approximately 42mA. This is within the 100mA maximum current rating of the 78L05 5V voltage regulator.

[14] Pin numbers as referenced in the Arduino programming environment.
[15] Fuse tutorial (based on ATmega328P): http://www.martyncurrey.com/arduino-atmega-328p-fuse-settings/

The alternative magneto-resistive sensor draws only approximately 7mA, compared to the approximately 25mA of the standard infra-red sensor.

### Microcontroller Package Ratings

The Atmel ATtiny85 microcontroller has an absolute source/sink current limit of 40mA per pin, and a total device package current limit of 200mA. The recommended total IO port current is 60mA. The sensor input pin was measured to sink approximately 130 µA when triggered. Other pins drive the diagnostic LED and the RS-232 line driver, and in total these do not draw more than a few milliamps, so the microcontroller limits do not present a problem.

### 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 One Bell 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[16]. For example, 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 42mA, this would result in a voltage drop of 0.17V along the 24m of cable, and therefore a theoretical minimum supply voltage to the cable of 7.87V 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 25m. (Cable voltage drop has a greater impact on larger Sensor Interfaces with large numbers of Sensor Heads.)

### Heat Dissipation

At peak load, with a current requirement of 42mA, the voltage regulator would be dissipating a minimum[17] of approximately 84mW. The dissipation will be higher with shorter cable runs or a higher supply voltage; for example with a 9V supply the dissipation would be up to 130mW. The data sheet for the voltage regulator indicates that this will not require a heatsink.

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

# Compatibility

## Hardware Compatibility

The One Bell 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.

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 Simulator Data Connection connector of the One Bell 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 One Bell Simulator Interface has been tested successfully with the following Simulator Software Packages:

- Abel 3.9.1
- Beltower 2015 (12.29)
- Virtual Belfry[18] 3.1b

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

[18] 3.1b is the minimum version of Virtual Belfry required for proper handling of interfaces of this type.

# Connector Pin-Outs

## Simulator Power/Data Connector

The One Bell Simulator Interface uses a GX16-5 (also known as “aviation”) connector for the cabling connection to the Simulator PC. This is the same connector used by the main version of the Simulator Interface.

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

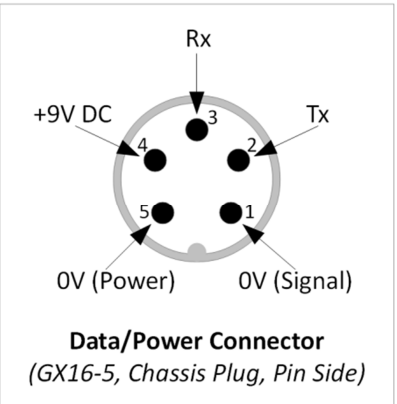



Figure 2 – Interface Data Connector

# Construction

The following sections describe the construction of the One Bell Simulator Interface.

## Simulator Interface Board

The One Bell Simulator Interface Board contains the power supply for the interface and sensor, the microcontroller, a RS-232 serial line driver, a diagnostic LED, and an ICSP[19] 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/onebellinterface
- https://oshpark.com/shared_projects/0aGxeqyO

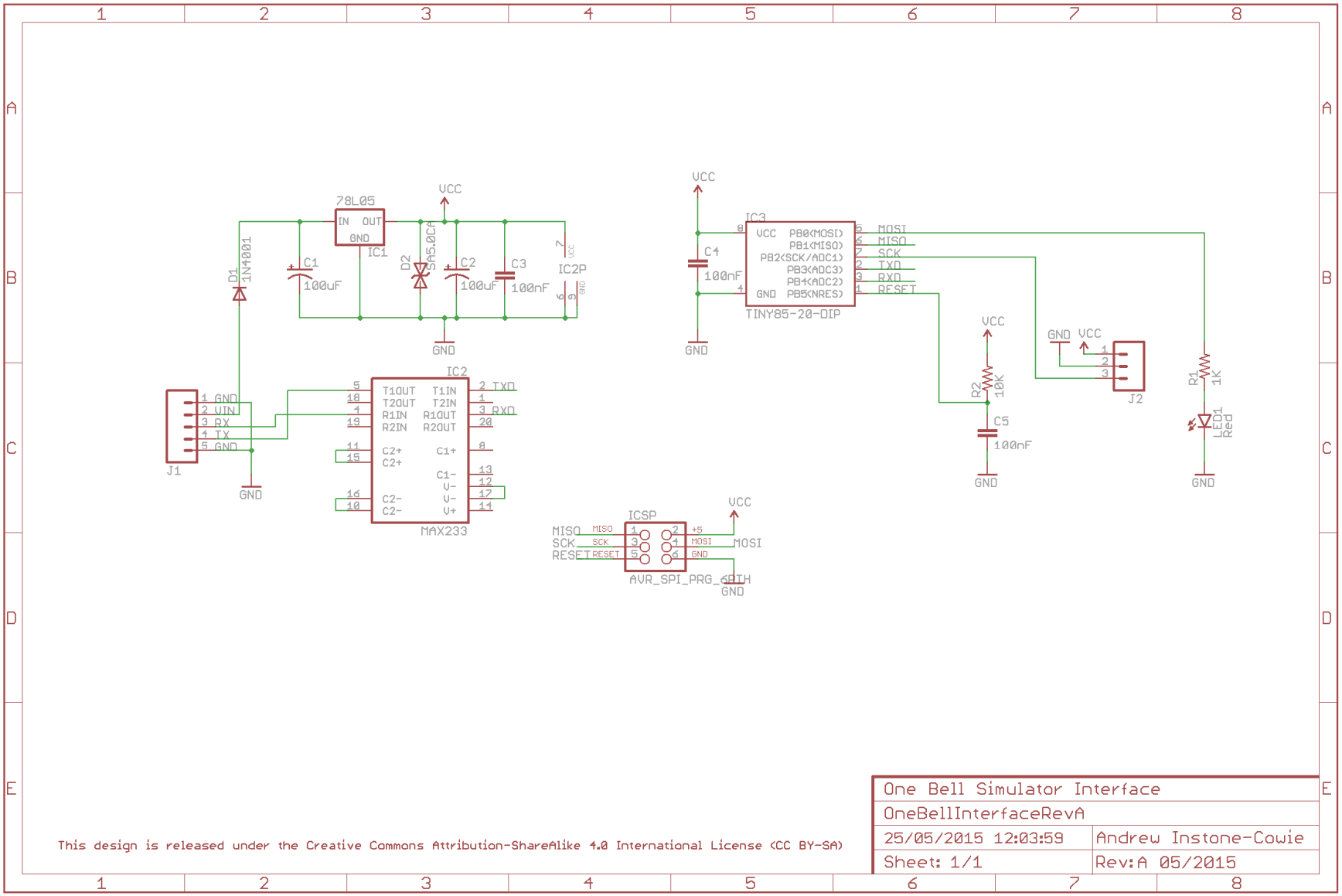
### Parts List

**Table 4 – One Bell Simulator Interface PCB Parts List**

| Reference | Component | Notes |
|---|---|---|
| PCB | One Bell Simulator Interface PCB – Rev A | https://oshpark.com/shared_projects/0aGxeqyO |
| R1 | 1kΩ 0.6W Metal Film | |
| R2 | 10kΩ 0.6W Metal Film | |
| C1, C2 | 100µF 25V Electrolytic (5mm Radial) | |
| C3, C4, C5 | 100nF 50V MLCC (2.54mm Radial) | |
| D1 | 1N4001 | |
| D2 | SA5.0 CA | Transient voltage suppression diode |
| IC1 | LM78L05 | TO-92 Case |
| IC2 | MAX233EPP | |
| IC3 | ATtiny85 | |
| LED1 | 3mm LED Red | Optional – see text |
| PC Header | 5-Pin 0.1” Male Header | |
| ICSP Header | 2x3-pin 0.1” Male Header | |
| Sensor Header | 3-Pin 0.1” Male Header | See text |
| IC Socket | 20-pin, 0.3” pitch | IC2 |
| IC Socket | 8-pin, 0.3” pitch | IC3 |
| Sensor | E18-D80NK Infra-Red Sensor | eBay |

[19] In-Circuit Serial Programming

## Schematic


78L05
IN OUT
GND
IC1
VCC
D1
1N4001
C1
100uF
D2
SA5.0CA
C2
100uF
C3
100nF
IC2P
GND
IC2
T1OUT T1IN
T2OUT T2IN
R1IN R1OUT
R2IN R2OUT
C2+
C1+
C1-
V-
C2-
V+
MAX233
J1
1 GND
2 VIN
3 RX
4 TX
5 GND
TXD
RXD
IC3
VCC PB0(MOSI)
PB1(MISO)
PB2(SCK/ADC1)
PB3(ADC3)
PB4(ADC2)
GND PB5(NRES)
TINY85-20-DIP
C4
100nF
MOSI
MISO
SCK
TXD
RXD
RESET
R2
10K
C5
100nF
J2
R1
1K
LED1
Red
ICSP
MISO
SCK
RESET
+5
MOSI
GND
AVR_SPI_PRG_6PTH
This design is released under the Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA)
One Bell Simulator Interface
OneBellInterfaceRevA
25/05/2015 12:03:59
Andrew Instone-Cowie
Sheet: 1/1
Rev:A 05/2015

## PCB Layout

The following diagram shows the layout of the One Bell Simulator Interface PCB.

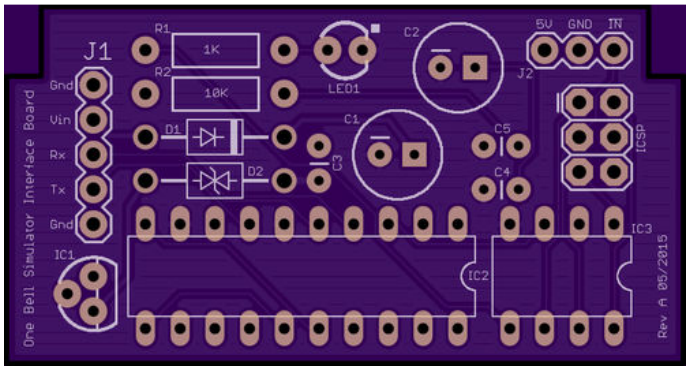



Figure 3 – One Bell Simulator Interface Board Layout

## Construction

For use with the standard infra-red sensor, all the components on the One Bell 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 electrolytic capacitors and header pins.
- The use of IC sockets for IC2 & IC3 is optional, but strongly recommended.
- Fitting the diagnostic LED is optional, but strongly recommended.
- 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 LED. This is normal if the firmware has not yet been uploaded to the microcontroller.
- Note that if the board is to be used with the alternative stackable magneto-resistive sensor board, then the sensor pins should be mounted on the reverse of the interface board, as illustrated below.

A completed One Bell Simulator Interface PCB is shown in the following photograph.

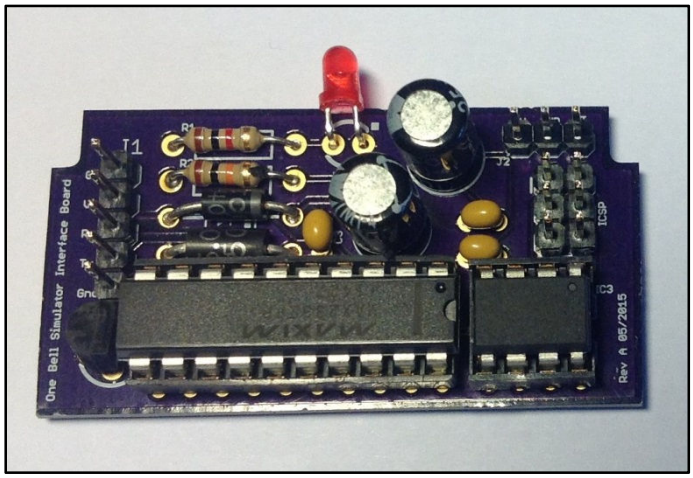



Figure 4 – Completed One Bell Simulator Interface PCB

## Enclosure

The suggested enclosure for the One Bell Simulator Interface is the WCAH 2855 black ABS box manufactured by Wisher Enterprise Co Ltd.

- This enclosure can be obtained from Maplin and ESR Components, and is the same enclosure adopted as a standard for Sensor Heads across the Liverpool Ringing Simulator Project.
- The enclosure is 85mm x 55mm x 30mm (external) overall.

### Parts List

Table 5 – One Bell Simulator Interface Enclosure Parts List

| Component | Notes |
|---|---|
| WCAH 2855 ABS Black Enclosure | ESR Components 400-595, Maplin BZ72P |
| No2 x 9mm Stainless Steel Self Tapping Screws | eBay |
| 30mm x 20mm (Diameter) Black Plastic Conduit | |

### Standard Infra-Red Sensor Construction

The following diagram gives a suggested drilling layout for the enclosure.

- Drilling large diameter holes with twist drills can result in bit grabbing and damage to the enclosure. Small diameter (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.
- Using a sharp chisel, carefully and gently remove the PCB ribs from the inside of the enclosure as indicated on the diagram below, to allow the Sensor and Power/Data connector to be mounted. Using too much force is likely to crack the case.
- It is recommended that the lid retaining screws supplied with the enclosure are replaced with stainless steel equivalents.

### *WCAH 2855 Enclosure*


Figure 5 – Enclosure Drilling Guide

### *PCB Mounting*

The One Bell Simulator Interface PCB is designed to be mounted in the uppermost set of PCB ribs inside the enclosure, as illustrated in the diagram above.

### *Sensor Mounting*

The standard infra-red sensor is mounted through the side of the enclosure using the plastic nuts supplied with the sensor. These should be tightened finger-tight only; do not use tools.

Once the sensor is installed, lightly file or sand the exposed threads so that the masking tube is a firm tight push fit on the end of the sensor.

The following photograph shows the PCB and Sensor mounted in the enclosure, and the internal cabling of the unit.

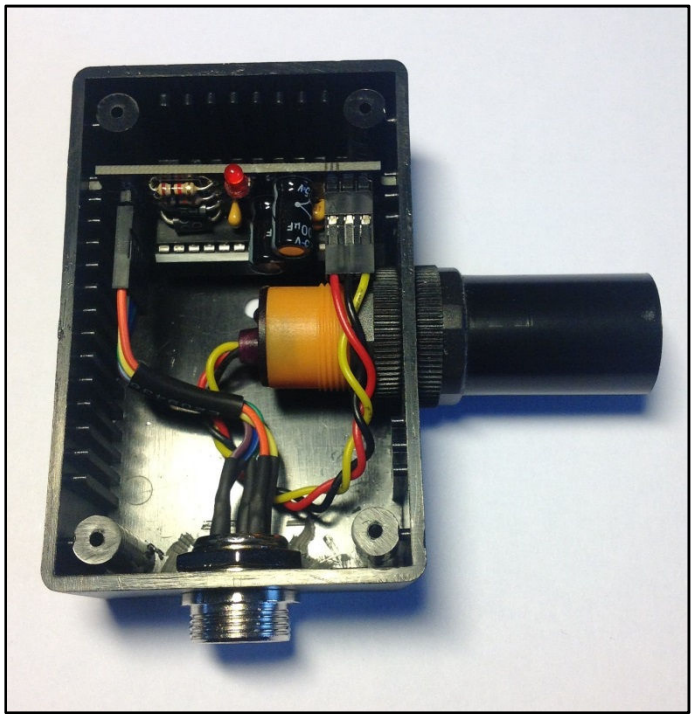


Figure 6 – One Bell Simulator Interface PCB & Sensor Mounting

## Alternative Magneto-Resistive Sensor Construction

As an alternative to the standard infra-red sensor, a design for an updated version (Revision C) of the magneto-resistive sensor PCB is available. This is described in detail in the Simulator Sensors Hardware Manual, and Eagle CAD and Gerber files for the board can be downloaded from GitHub.

This version of the magneto-resistive sensor PCB is designed to fit into the same enclosure as the One Bell Simulator Interface PCB, and connect to it by means of a 3-pin stacking header. The sensor pins on the interface PCB are mounted on the reverse of the board, and mate with a connector on the sensor PCB. The following photograph shows the stacked boards.

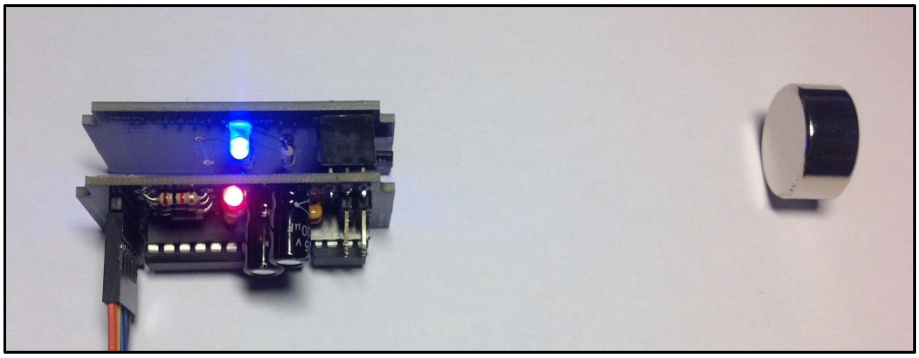


Figure 7 – One Bell Interface & Magneto-Resistive Sensor Boards Stacked

### *PCB Mounting*

The stacked pair of PCBs can be slotted into the PCB mounting ribs of the project standard WCAH 2855 enclosure, as shows in the following photograph.

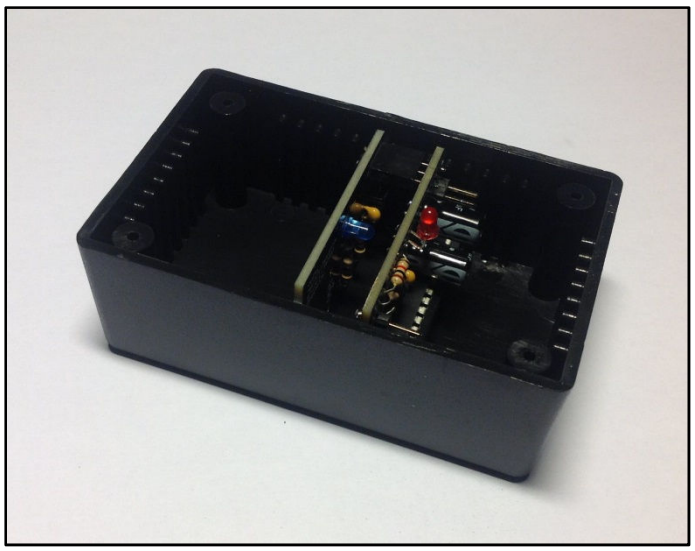


Figure 8 – Enclosure with Stacked Interface & Sensor PCBs

## Internal Cabling

Two internal cables are required within the One Bell Simulator Interface enclosure, to connect the interface board to the external power/data connector, and to connect the infra-red sensor to the interface board:

- Sensor Connector Cable (standard infra-red sensor only)
- Power/Data Cable

### Parts List

Table 6 – Internal Cabling Parts List

| Component | Notes |
|---|---|
| GX16-5 Chassis Plug | Power/Data Connector |
| Heat Shrink Sleeving 2.5mm Un-shrunk Size | |
| Ribbon Cable | |
| DuPont 0.1” Female Crimp Connectors | 8 |
| DuPont 0.1” Female Connector Shells | 1 x 5-pin, 1 x 3-pin |

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.

## Standard Sensor Connector Cable

The Sensor Connector Cable is integral to the standard infra-red sensor. This cable should be shortened to approximately 6cm, and a 3-pin female DuPont connector crimped to the end, wired as shown the following diagram.

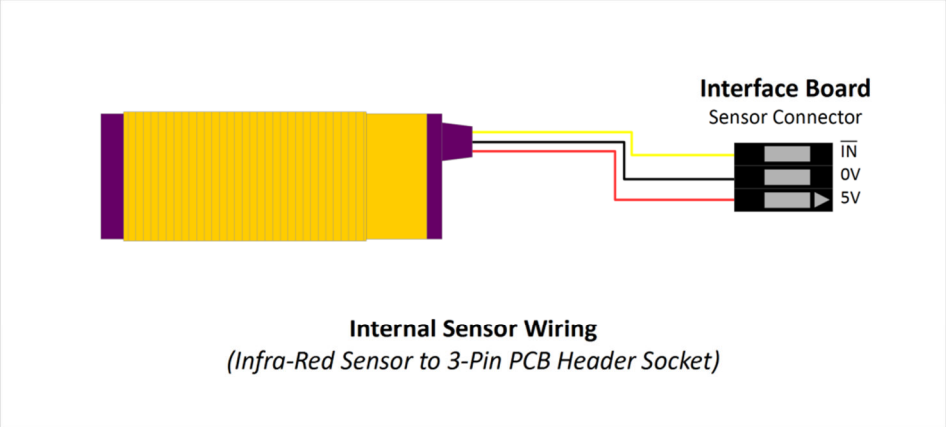



Figure 9 – Internal Sensor Cable Diagram

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.

## 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.

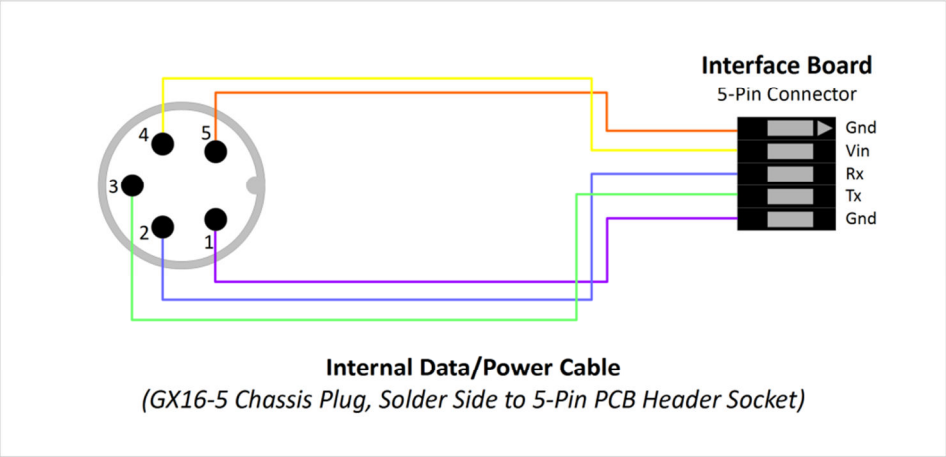



Figure 10 – Internal Power/Data Cable Diagram

A suitable length of ribbon cable can be used for this cable, the required length of the cable is approximately 6cm 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 Cable is shown in the following photograph. Note that this is a test cable and much longer than required. Another example can be seen in the photograph above.

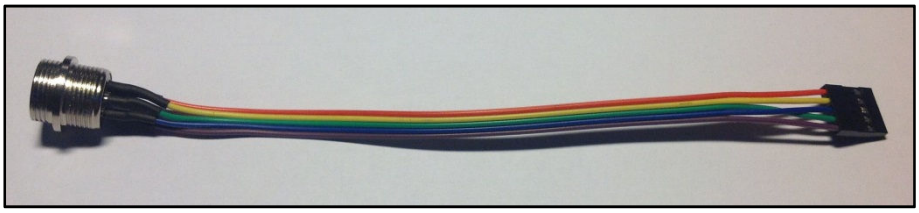


Figure 11 – Power/Data Connector Cable

## External Cabling

A Power/Data external cable is required to interconnect the One Bell Simulator Interface to the Simulator PC. This cable is described below.

### Parts List

Table 7 – External Cabling Parts List

| Reference | Component | Notes |
|---|---|---|
| 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 | |

## Power/Data Cable

A single Power/Data Cable is required to connect the One Bell 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.

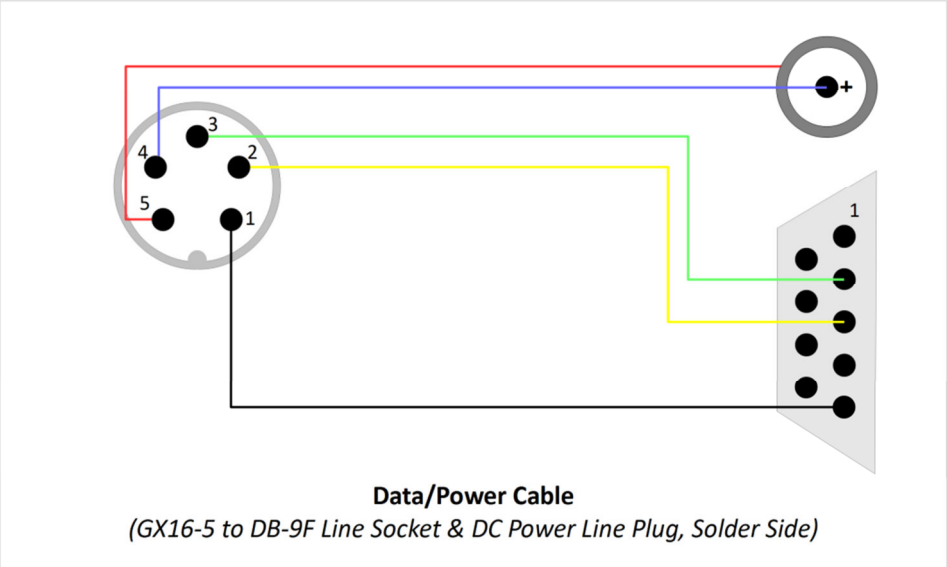



Figure 12 – Simulator Data Cable Wiring

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 8 – 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.

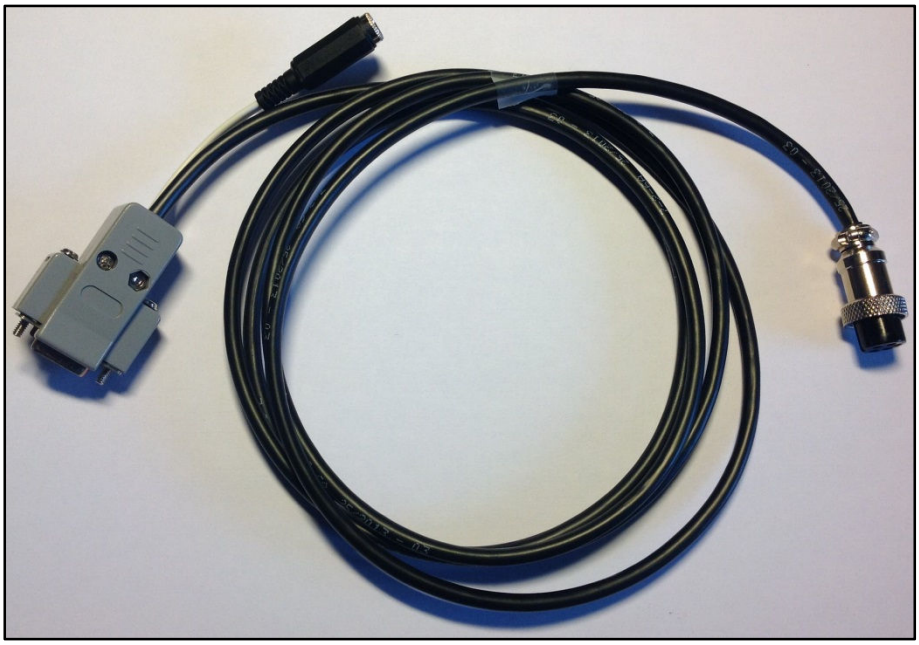


Figure 13 – Power/Data Cable

## 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 the Sensor connected.
- The supply voltage may be higher than that required to maintain 7.5 volts at the Interface, but this may result in overheating of the voltage regulator IC1 if the input voltage is excessive.
- As a guideline, a supply voltage of 9V has been found sufficient to maintain the required voltage with a 25m Power/Data cable.

## Firmware Upload

The firmware for the One Bell 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/onebellinterface

The One Bell Simulator Interface firmware is held in non-volatile flash memory on the ATtiny85 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 One Bell Simulator Interface PCB. Although the software development environment (described in the Software Manual) is based on the Arduino platform, the One Bell 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*[20].
- An Arduino add-on board or shield such as the *Arduino ISP*[21] or the BoardStuff *UNO Multi Programming Shield*[22].
- 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[23].

[20] http://www.atmel.com/tools/avrispmkii.aspx
[21] http://www.arduino.cc/en/Main/ArduinoISP
[22] http://www.boardstuff.co.uk/
[23] 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 One Bell Simulator Interface firmware:

- Download and install the latest Arduino IDE package[24]. At the time of writing this was version 1.6.3.
- The Arduino IDE does not support the ATtiny series of microcontrollers as standard. Download the ATtiny board support package from: https://github.com/damellis/attiny/tree/ide-1.6.x (use the *Download ZIP* facility on GitHub).
- Start the IDE, and locate the Arduino sketchbook directory by selecting *File | Preferences*.

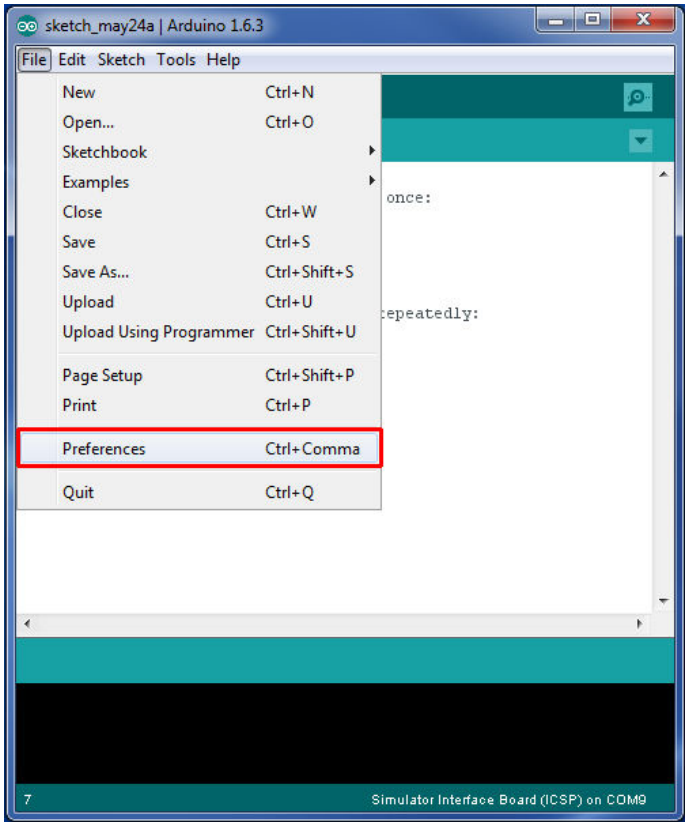



Figure 14 – Arduino IDE Preferences Menu

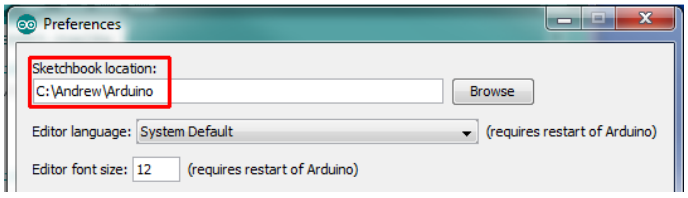



Figure 15 – Arduino IDE Sketchbook Location

- Unzip the downloaded ATtiny support package, and place the `attiny` directory into a directory called `hardware` within the sketchbook directory, so that the path looks like this: `hardware\attiny\avr\variants\...`

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

- In the `avr` subdirectory, append the following lines to the text file called `boards.txt`. This file defines the parameters for uploading to the One Bell 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:

```
# One Bell Simulator Interface Board - HW Rev A
onebell.name=One Bell Simulator Interface Board
onebell.bootloader.tool=arduino:avrdude
onebell.bootloader.unlock_bits=0xff
onebell.bootloader.lock_bits=0xff
onebell.build.core=arduino:arduino
onebell.build.board=attiny
onebell.upload.tool=arduino:avrdude
onebell.upload.maximum_size=8192
onebell.build.mcu=attiny85
onebell.build.variant=tiny8
onebell.bootloader.low_fuses=0xe2
onebell.bootloader.high_fuses=0xd7
onebell.bootloader.extended_fuses=0xff
onebell.build.f_cpu=8000000L
```

- Close and 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 16 – Arduino USB Cable

The Arduino website has instructions[25] 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.

[25] 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.

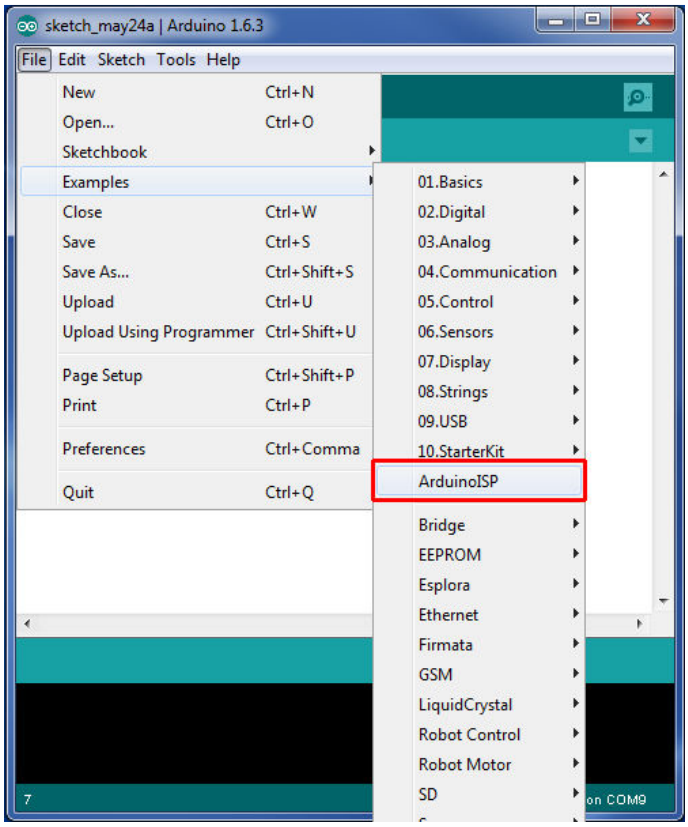



Figure 17 – Arduino IDE ISP Sketch Loading

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

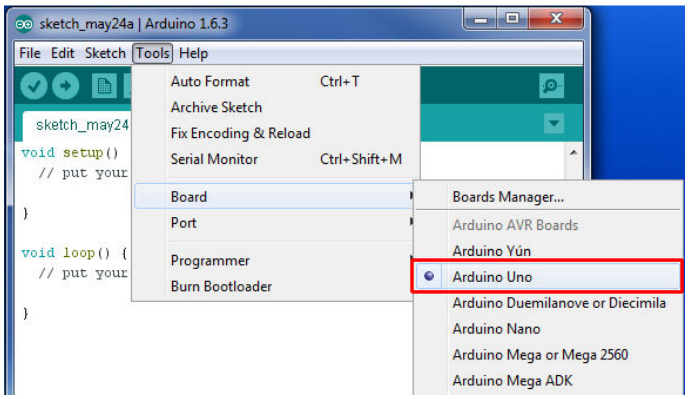



Figure 18 – Arduino Programmer Board Selection

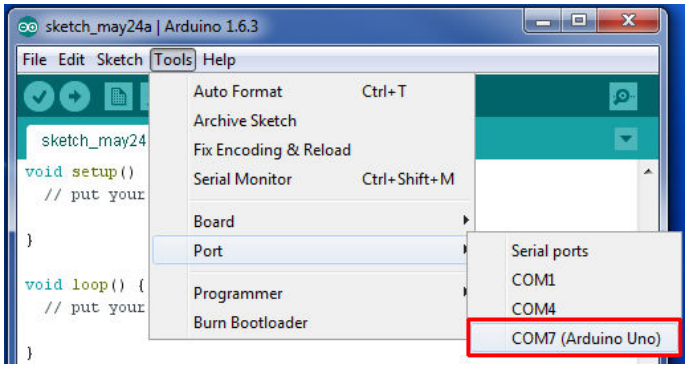



Figure 19 – Arduino Programmer Port Selection

- 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.

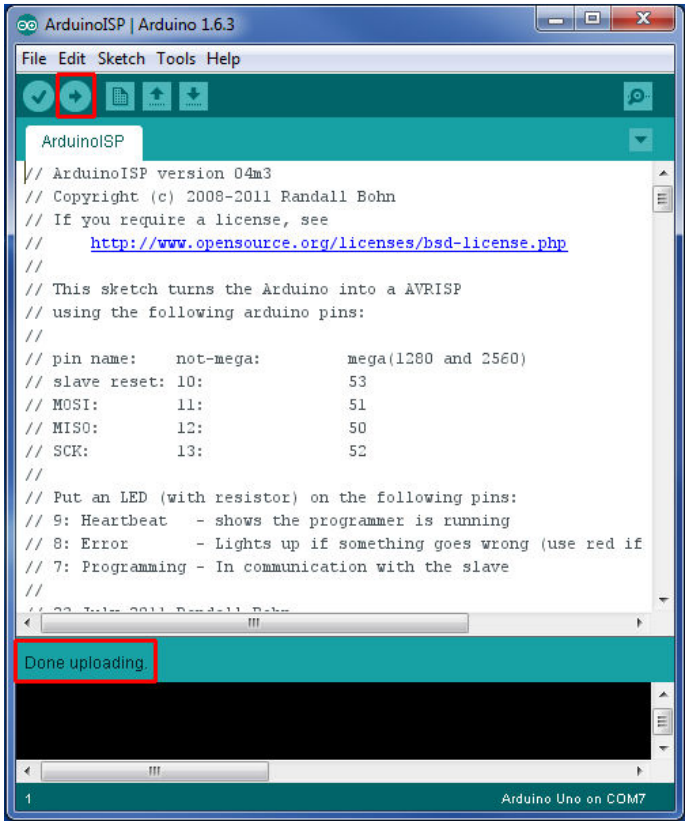



Figure 20 – Arduino IDE ISP Upload

- 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.

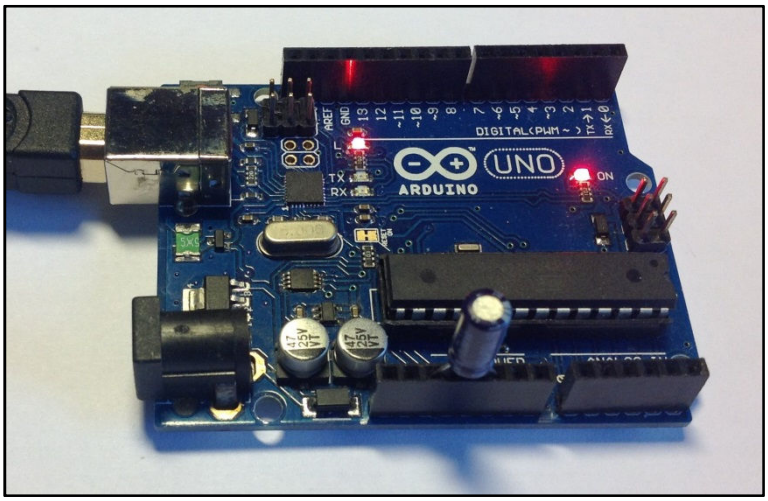



Figure 21 – Programmer with Capacitor

- 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.
- Disconnect the sensor from the interface PCB.
- 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 One Bell Simulator Interface PCB is top left, identified by a white stripe.
- Pin 1 on the programmer is top left. Note that pin 5 on the One Bell Simulator Interface PCB is connected to pin 10 on the programmer.

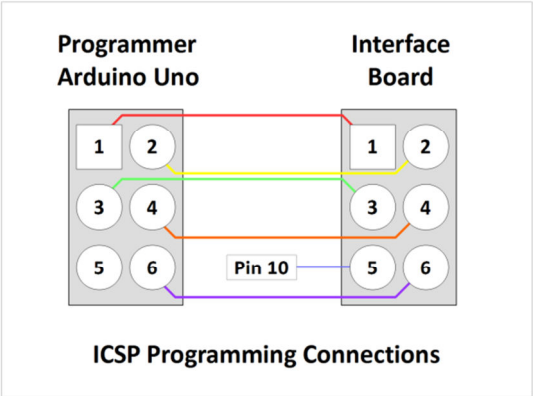



Figure 22 – Programmer Connections

- 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.

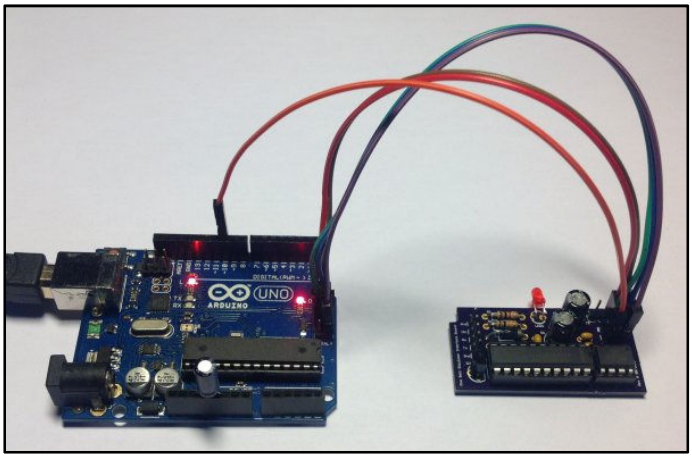


Figure 23 – Programmer Connected to One Bell Simulator Interface Board

- 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 *One Bell Simulator Interface Board*.

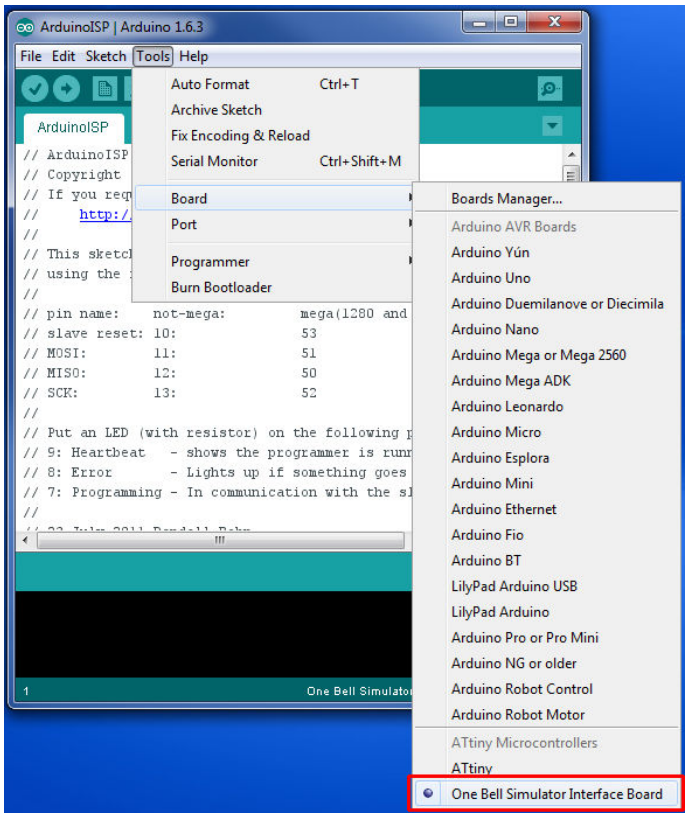



Figure 24 – Arduino IDE Target Board Selection

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

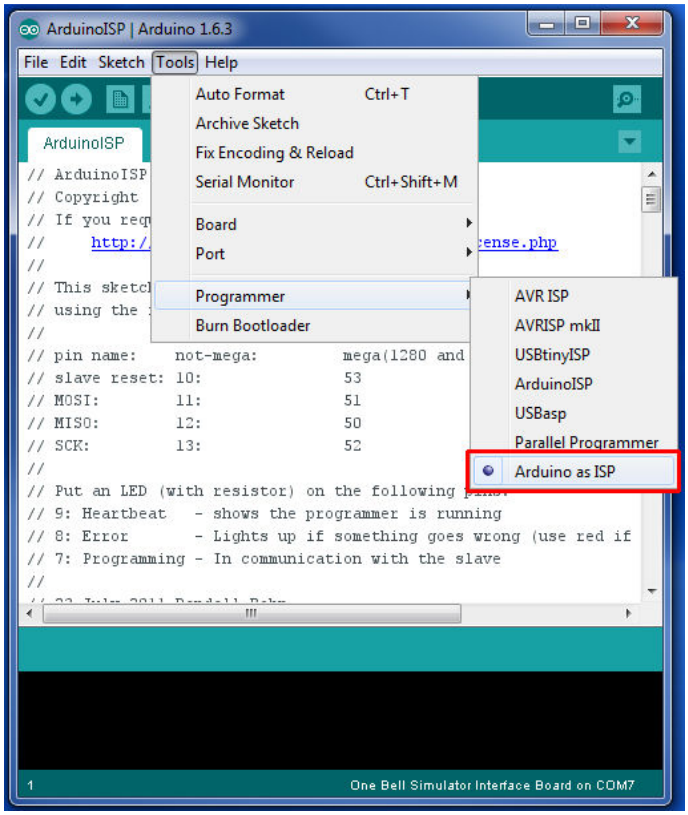



Figure 25 – Arduino IDE Target Board Selection

- On the *Tools* menu, select *Burn Bootloader*. No bootloader will be written, but this step is necessary to program the microcontroller fuses. Verify that the burn process completed successfully by looking for the *Done burning bootloader* message.

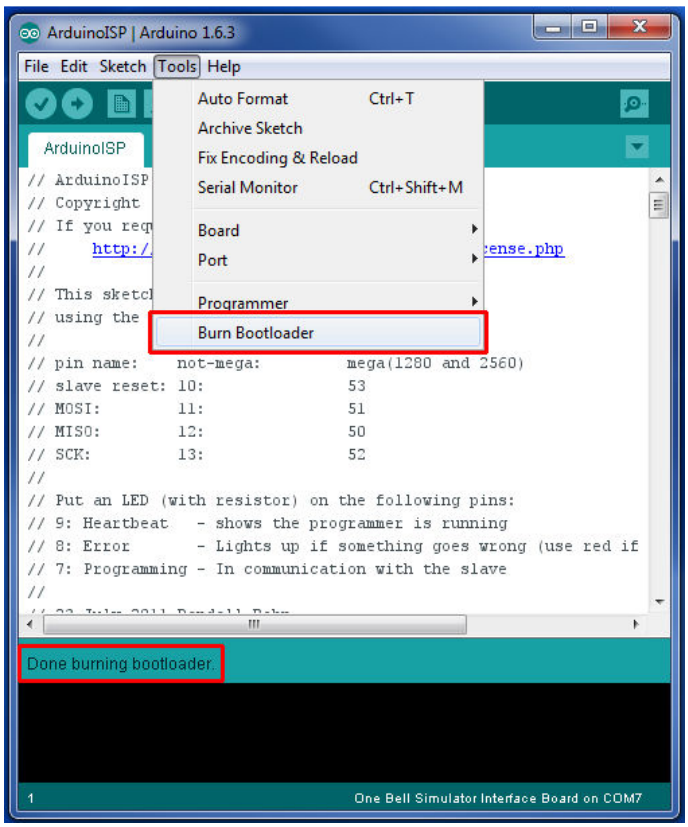



Figure 26 – Arduino IDE Burn Bootloader

- Note that if new firmware is being uploaded to an existing Simulator Interface Board, there should be no need to go through the above 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 One Bell Simulator Interface firmware to the board.

- Connect the Simulator Interface Board to the programmer as described in the previous section.
- Download the One Bell Simulator Interface firmware and load it into the Arduino IDE by double clicking the name of the main file, e.g. *OneBellInterface_v1_1.ino*.
- On the *Tools | Board* menu, as above ensure that the correct board type to be programmed has been selected, in this case *One Bell Simulator Interface Board*.
- 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.

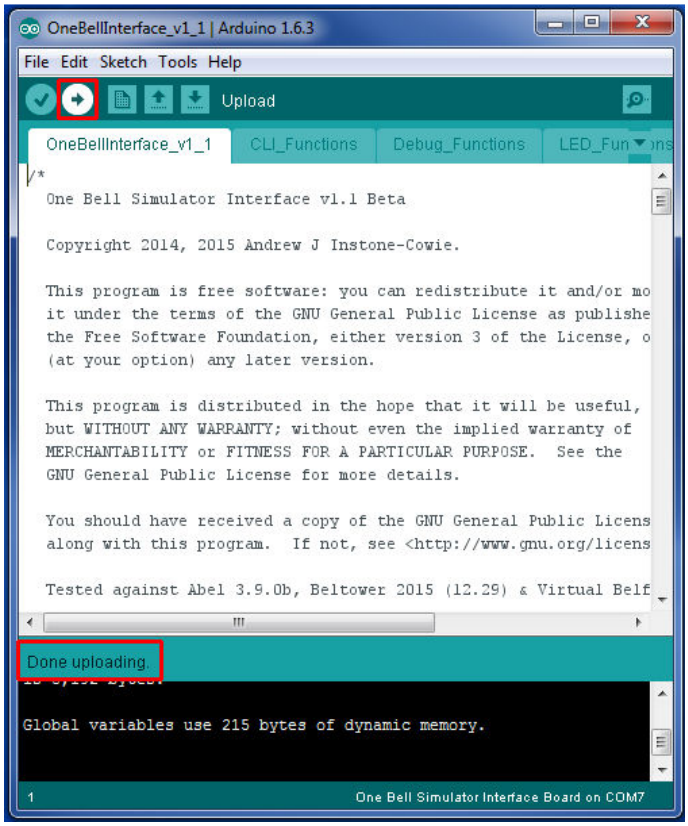



Figure 27 – Arduino IDE Firmware Upload

- 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 One Bell Simulator Interface board will be reset, and on restarting the red diagnostic LED will flash according to the firmware version, for example one long and one short flashes indicates firmware version 1.1.
- Disconnect the USB cable from the programmer.
- Disconnect the programmer from the One Bell Simulator Interface Board.
- Note that when uploading new firmware to an existing One Bell Simulator Interface Board, the Sensor and the Power/Data Cable must be disconnected from the One Bell Simulator Interface.

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.

- Fit the Power/Data Connector Cable assembly to the end of the enclosure and tighten the securing nut.
- Connect all the cables to their respective pins on the One Bell Simulator Interface Board, and carefully fit the lid to the enclosure using the screws supplied.

An example of a completed One Bell Simulator Interface is shown in the following photograph.

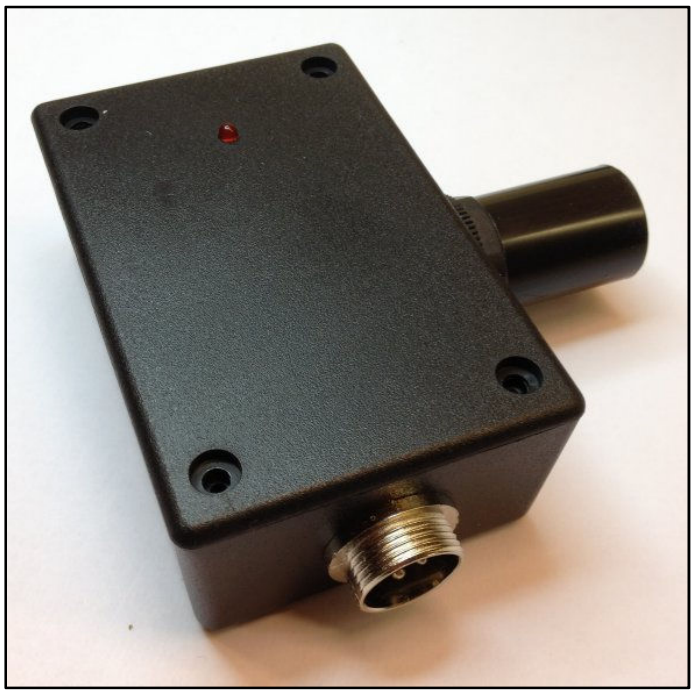


Figure 28 – Completed One Bell Simulator Interface

# Installation

## Simulator Interface

The One Bell Simulator Interface is located in the belfry, attached to the frame in a location such that the sensor tube is perpendicular to the face of the shroud of the wheel, in the same manner as a standard Sensor Head.

Figure 29 – Example of an Installed Simulator Sensor Head

### Reflector

The sensor requires a reflector mounted on the shroud of the bell wheel, such that the reflector is opposite the Sensor Head when the bell is at the bottom of its swing. Consequently it is easiest to fit the reflector when the bell is down.

The reflector is made from a short length of white reflective automotive styling tape, 25mm wide. This may be obtained from a car spares shop. This can be seen in the photograph above.

### Calibration

As supplied, the infra-red detector sensor modules were found to draw approximately 55 – 60mA, much more than the specified 25mA, and were excessively sensitive. The small calibration screw on the end of the module may be used to reduce both the current consumption and sensitivity of the detector.

A useful starting point for sensitivity adjustment has been found to be to reduce the sensitivity of the sensor such that it does not trigger when placed perpendicular to a piece of grey card at a distance of 90mm. The multi-turn adjustment screw is turned anti-clockwise until the indicator LED on the back of the module just goes out. This gives an effective trigger distance with the reflective tape of about 300mm. This also reduces the supply current to approximately 21mA untriggered, 28mA triggered, although this is variable between modules.

The sensor should then be adjusted in the belfry for optimum sensitivity.

## Simulator Power/Data Cable

The Simulator Power/Data Cable is routed from the One Bell 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.