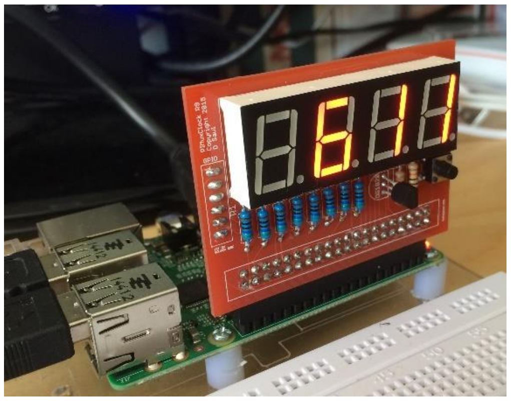
# PiMuxClock

The PiMuxClock is a simple Raspberry Pi add-on card, allowing you to add a basic digital clock and temperature display for your RPi. The key design goal for the PiMuxClock was to get back to basics and develop a low cost, educational hardware project. No buffers or serial interfaces just direct connection to the Raspberry Pi GPIO, programmable with simple Python code to drive a multiplexed display.



## 1 Introduction – How to use this document

This document contains all should need to build / get working / develop your PiMuxClock. For those with ready built boards, who just want to get something working quickly you can jump to the section 2 'quick start'.

The document is laid out in what you will hopefully find is a logical order starting with build instructions. The build is straightforward but there is the potential to solder parts in the wrong place so please don't dive straight in with your soldering iron without at least glancing at the instructions first.

These instructions assume you are comfortable with basic operation of the Raspberry Pi and the Raspbian operating system. If you are new to the Raspberry Pi you should get yourself familiar with its operation before attempting to connect and operate your PiMuxClock. There are lots of beginner's guides on the internet, the official Raspberry Pi foundation material can be found at www.raspberrypi.org/help/ .

If you are considering the PiMuxClock for educational purposes, lesson plans and additional activity support material is available with the educational PiMuxClock packs. These can also be adapted to suit specific needs if required.

### 1.1 Do's and don'ts

The PiMuxClock connects directly to the Raspberry Pi's GPIO. The Pi's GPIO lines are not buffered so shorting them or applying voltages outside the specified limits could result in damage to the Raspberry Pi. For more information on the Pi's GPIO please follow this LINK to the official Raspberry Pi site.

PiMuxClock is intended as an electronics / software development project, in connecting your PiMuxClock pledge reward to a Raspberry Pi computer you are accepting that the supplier / designer of the PiMuxClock cannot be held liable for any consequential damage or losses how so ever caused.

The PiMuxClock has been tested extensively with the Pi A+, B+ and Pi B Model 2. It is not suitable for use with the original A or B versions, these are easily identified as they have the smaller 26 way GPIO connector.

All instructions and design testing have based on the default BCM2835/6 'drive strength' setting. This is low level configuration element that you should not attempt to change under any circumstance, doing so could result in damage to your Pi.

# Contents

1 Introduction – How to use this document
2 Quick start
3 Build Instructions
4 Connecting the PiMuxClock to your Raspberry Pi
5 Raspberry Pi Setup
6 Supplied Sample Software
7 Using your PiMuxClock
8 Theory of Operation
9 Project Extension ideas
10 Problems
11 Appendix, Technical Spec Information

# 2 Quick start

Ok I know you just want to see your PiMuxClock working but, before we start please to take a moment to read through the do's & don'ts on page 1.

This section assumes you have a built PiMuxClock board and simply takes you through how to get your PiMuxClock working in the basic clock mode, it does not cover the enhanced functionality.

## 2.1 Connecting your PiMuxClock

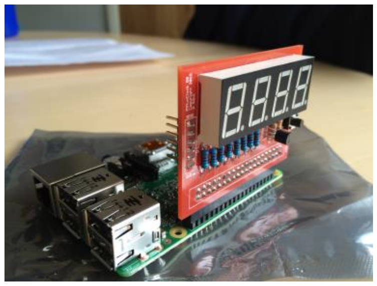


*Figure 1 board orientation*

- Ensure your Raspberry Pi is connected to the internet
- Download the file *basicclock_2_r1.py* from www.pimuxclock.co.uk to your Pi (for more on downloading software see section 6.1)
- Shutdown your Pi and disconnect the power supply
- Ensure there is nothing else connected to the GPIO connector
- Carefully insert the PiMuxClock board into the GPIO connector so the board front is facing away from the Pi – see figure 1, making sure that board and Pi connectors are correctly aligned.
- Reconnect power to the Pi
- When it has booted up navigate to the directory you downloaded the *basicclock_2_r1.py* file to
- Start the program by entering *sudo python basicclock_2_r1.py*
- The clock should start immediately. Depending on which version of the Raspberry Pi you are using you may notice some display flicker, this is normal – see section 6.2

## 3 Build Instructions

If you opted for the bare PCB pledge award the full parts list can be found in the appendix [11.3].

**Preparation**

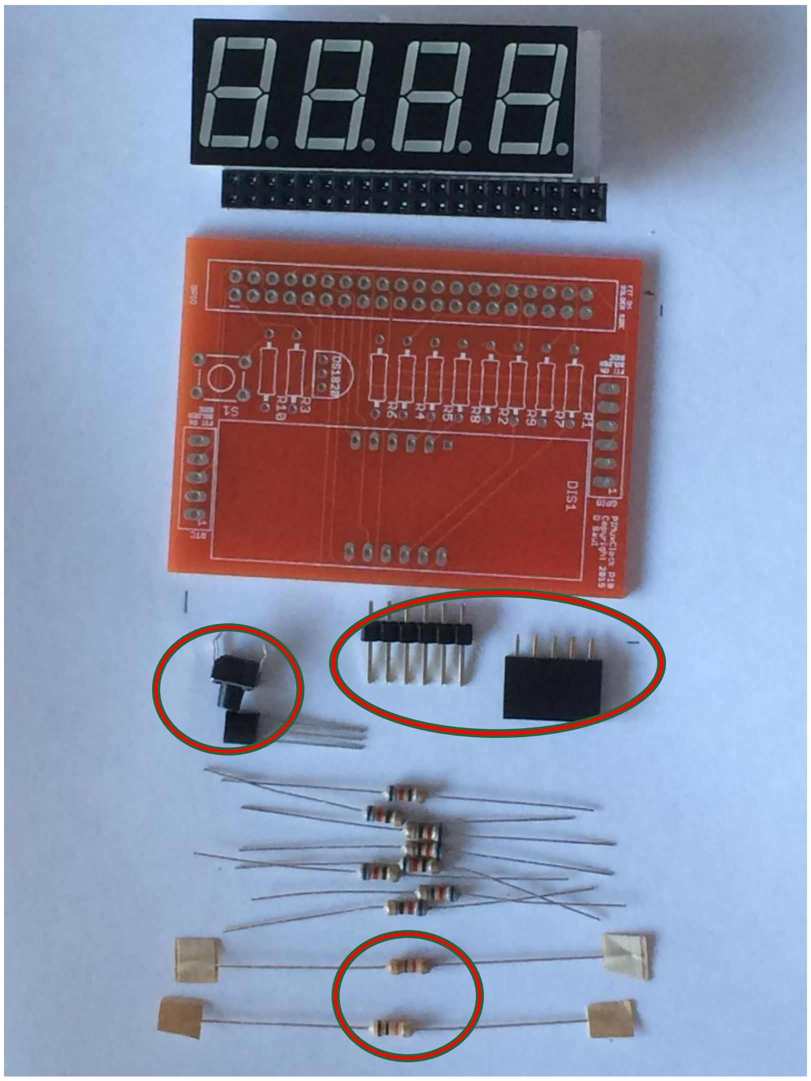
Please read through all the instructions before you start. The instructions assume a basic level of soldering competence, if you are in any doubt there is lots of material on web to help you – one example on the Adafruit site is - adafruit-guide-excellent-soldering

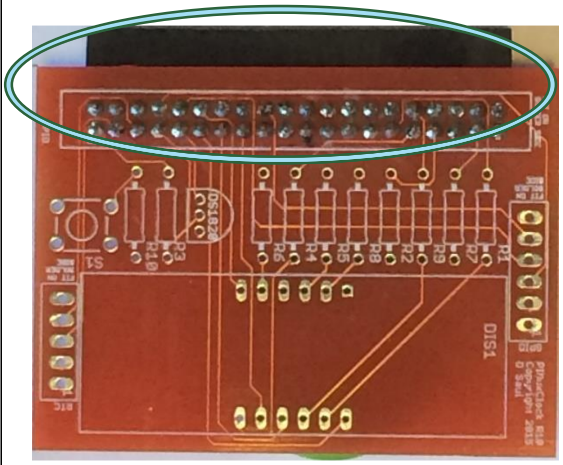
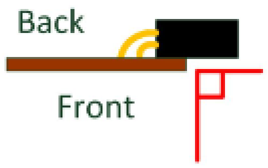
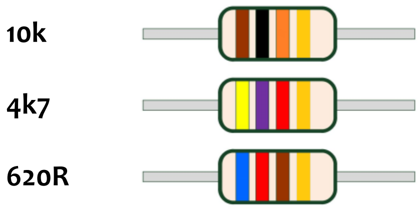
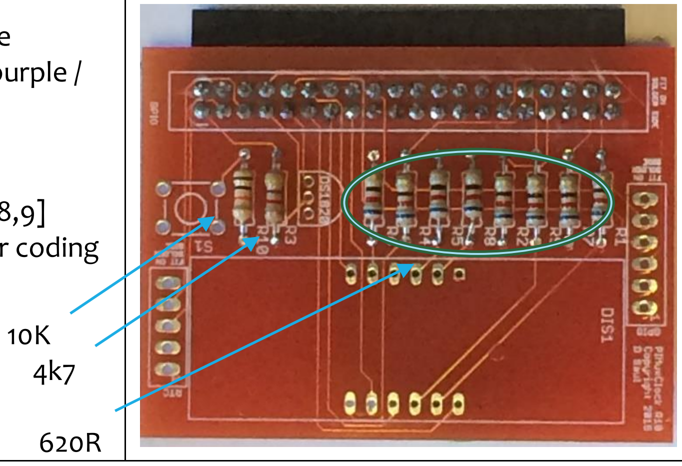
- There are no large areas of copper, so a 25W or less soldering iron will be fine
- When handling the DS18B20 IC ensure you take static precautions
- Make sure you fit the connectors on the rear of the PCB
- Be careful to fit the resistors in correct place
- If you need to force anything you are trying to put it in the wrong place!

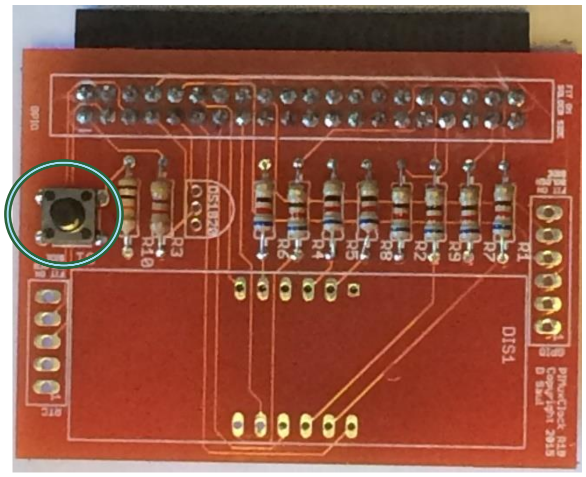
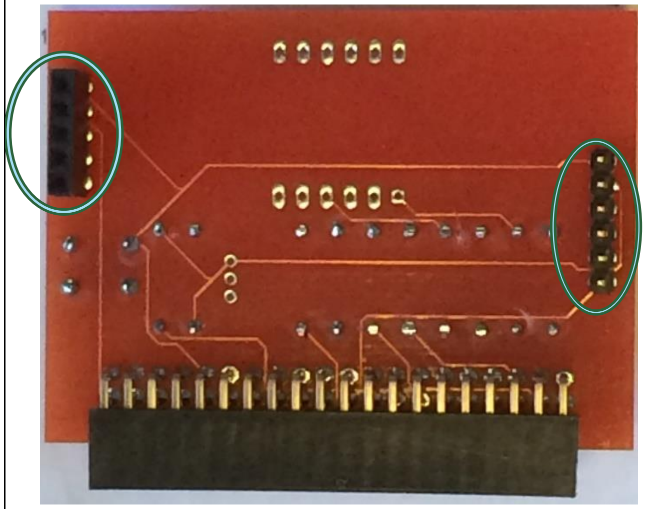
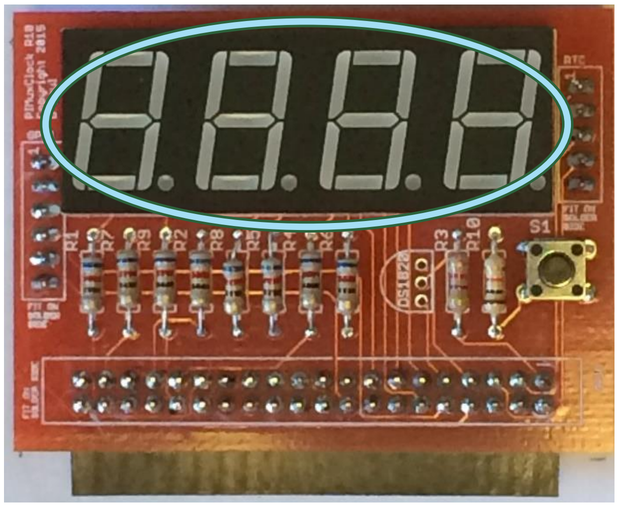
1 **Part Check**

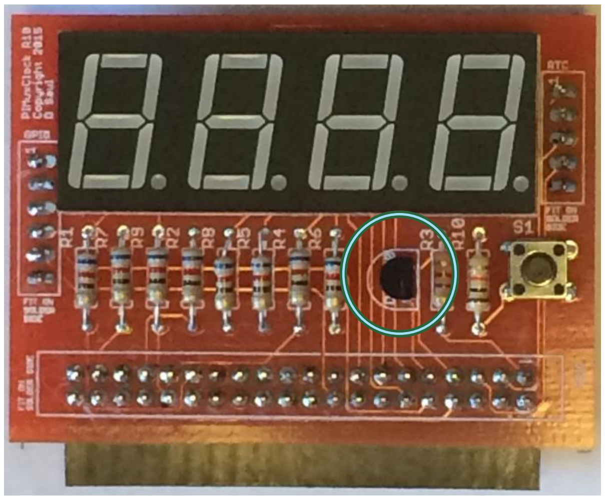
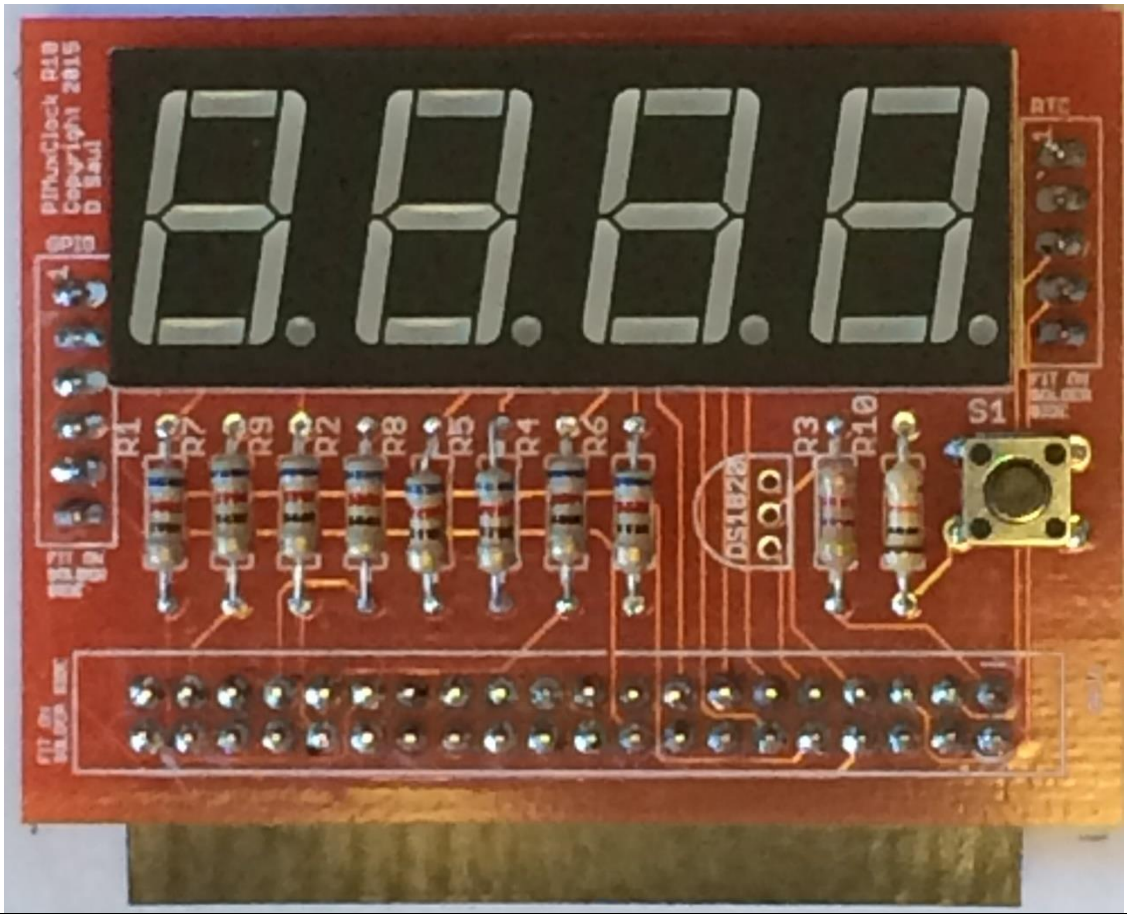
Lay all the parts out to confirm you have everything.

Note : - The circled items are only supplied with the enhanced kits



| | | |
|---|---|---|
| 2 | **40 way connector**<br><br>Solder the 40 way Pi GPIO connector to the **<u>REAR side of the PCB</u>**. To ensure the PiMuxClock stands perpendicular  to the Pi when complete it is important to make certain the connector is fitted square to the PCB [see diagram to right]. To do this firstly make sure the PCB is supported then solder a single corner pin first, check the alignment carefully re-soldering if needed. Then solder the opposite corner pin and again adjust the alignment by re-melting the solder if needed. Once you are happy that it is square solder the remaining pins. | <br><br> |
| 3 | **Resistors**<br><br>Fit and solder resistors;<br><br>**<u>Enhanced kits only</u>**<br>One  10k ohm resistor [R10] this can be identified by it's colour code brown / black / orange / gold<br><br>One  4k7 ohm resistor [R3] this can be identified by it's colour code yellow/purple / red / gold<br><br>**<u>All kits</u>**<br><br>Eight 620 ohm resistors [R1,2,4,5,6,7,8,9] these can be identified by their colour coding blue / red / brown / gold. | <br> <br><br> |

|  |  |  |
|---|---|---|
| 4 | **Tactile switch - <u>Enhanced kits only</u>**<br><br>Fit and solder in the switch, again being carful that it is square to the board |  |
| 5 | **Rear Connectors - <u>Enhanced kits only</u>**<br><br>Fit and solder the rear connectors to the <u>rear</u> side of the board, being careful that they are square to the board |  |
| 6 | **Display**<br><br>Fit and solder in the 4 digit LED display |  |

| | | |
|---|---|---|
| 7 | **DS18B20 - <u>Enhanced kits only</u>**<br><br>Taking static precautions fit and solder in the DS18B20 temperature sensor |  |
| 8 | **Visual check**<br><br>Visually check the completed board to check there are no solder shorts, dry joints or parts in the wrong place<br> PiMuxClock R10<br>Copyright 2015<br>D Saul<br>GPIO<br>RTC<br>R1<br>R7<br>R9<br>R2<br>R8<br>R5<br>R4<br>R6<br>R3<br>R10<br>S1<br>DS18B20 | |

# 4 Connecting the PiMuxClock to your Raspberry Pi

No modifications to your Pi board are needed to connect the PiMuxClock board, but obviously if it is in a box you will need to be able to access the full GPIO connector.

To connect the PiMuxClock;

- *Shutdown your Pi and disconnect the power supply*
- *Ensure there is nothing else connected to the GPIO connector*
- *Making sure the bottom the Pi is well supported, carefully insert the PiMuxClock board into the Pi GPIO connector so the board front is facing away from the Pi – see figure 1*
- *Make sure that board / Pi connectors are correctly aligned and that the PiMuxClock is fully seated into the Pi's GPIO connector*
- *Reconnect power to the Pi*

# 5 Raspberry Pi Setup

## 5.1 Assumptions

These the instructions assume you are using the *RASPBIAN* operating system as supplied in the *NOOBS install version 1.4 or later*. The latest image they have been tested against is version 1.4.1 dated 11th May 2015 [kernel 3.18].

There have been a number of incremental improvements / changes to the *RASPBIAN* operating system since its initial release. That said in its basic form the PiMuxClock should work with any version dating from the release of the B+ in July 14.

To keep everything simple the example software uses the default *RPi.GPIO* library to control the GPIO lines driving the multiplexed display.

The instructions are written on the basis that you have no other specialist GPIO functionality implemented.

As with all software nothing stays the same for very long. The plan is to provide updates these instructions via the website to cover any significant changes to *RASPBIAN* that could affect operation / setup of the PiMuxClock until at least mid 2016. It is however always a good idea to keep an eye on the Raspberry Pi forum for discussion on changes / developments to the operating system.

## 5.2 Basic

There is no additional system setup needed on your Pi for basic clock operation, go straight on to section 6.1, sample software.

## 5.3 Enhanced

Kernel updates in early 2015 to support the 'Device Tree' [DT] concept, changed the recommended setup of I2C and 1-wire interfaces. If you are still using an earlier version of *RASBIAN* you need to work to the legacy setup methods for 1W and I2C, rather than the following instructions.

### 5.3.1 Configuring the 1 Wire interface

The 1-wire interface is used to read the Ds18B20 temperature sensor fitted on the Enhanced PiMuxClock on GPIO 04 line, it does not need to be configured for the basic clock display.

Assumptions

- Clean NOOBS install [1.4 or above]
- The Raspberry Pi Connected to internet by cable or wireless
- You have connected the PiMuxClock board to your Pi [see section 4]
- You have not already enabled the 1-wire interface

From the console or a *terminal* window in the graph user interface [GUI] enter the following commands.

| | **Action** | ***Command(s) to enter*** |
|---|---|---|
| 1 | Ensure you are running latest s/w | *sudo apt-get update*<br>*sudo apt-get upgrade* |
| 2 | Download the '*seq_test*' programs | See 6.2.4 |
| 3 | Run *seq_test*, to confirm hardware is connected ok | *sudo python seq_test_2_r1.py* |
| 4 | Stop the application | hit '*control + c*' |
| 5 | Edit config.txt to allow DT support for the 1-wire interface to be enabled | *sudo nano /boot/config.txt* |
| 6 | Add '*overlay=w1-gpio*' command | At the end of the config.txt file add '*dtoverlay=w1-gpio*'<br>Save and exit the editor |
| 7 | Setup the 1-wire interface to start at power up, and restart | *sudo modprobe w1-gpio*<br>*sudo modprobe w1-therm*<br>*sudo reboot* |
| 8 | Check it is all working, after entering the 'ls' command you should see the following response<br>28-0000058d6e88  w1_bus_master1<br>**Note** the number following the 28- will be different. This is the DS18B20's id and is unique to an individual sensor | *cd /sys/bus/w1/devices*<br>*ls* |
| 9 | Change to the sensor sub-directory | *cd 28-xxxxxxxxxxx*<br>[replace xxxx with your specific sensors id ] |
| 10 | Check the readback<br>You should see something similar to this,<br>b0 01 4b 46 7f ff 10 10 3a : crc=3a YES<br>b0 01 4b 46 7f ff 10 10 3a t=27000<br>The 't' value is the measured temperature in thousands of the degree centigrade. The 'YES' indicates a valid reading | *cat w1_slave* |
| 11 | The 1-wire interface is now enabled and running on GPIO 04, it will auto start at power up | |

*Table 1, 1-wire setup*

### 5.3.2 Configuring the DS1307 RTC [I2C] interface

An RTC is only needed if you are wanting to use the PiMuxClock with a standalone Pi [i.e. without an internet connection], this is because with an internet connection the Pi will automatically set its internal software clock from the internet if available.

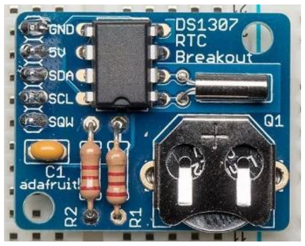


*Figure 2 Adafruit Ds1307 RTC*

Adafruit provide extensive instructions for configuring their 'DS1307 Real Time Clock breakout board kit' you can access these through the following links;

- *I2C setup*
- *Configuring the Hardware clock*

These instructions have been checked with the PiMuxClock board.

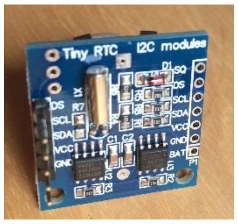


*Figure 3 'Tiny' RTC board*

There are a number of other DS1307 boards including the 'Tiny RTC' which should also work with PiMuxClock board, the main points to be careful are that you connect them correctly and ensure that no +5V pull-up resistor is fitted. The reason for this is explained in the Adafruit instruction links above.

Pin assignments for the PiMuxClock RTC interface can be found in the appendix [sec. 11].

# 6 Supplied Sample Software

## 6.1 Downloading PiMuxClock sample software

The sample software can be downloaded from the PiMuxClock website [www.pimuxclock.co.uk] - it can be found on the download page. To save the required files to your computer using the Pi's epiphany browser right-click on the filename and select '*save link as*'.

### Software naming convention

Example software is available for Python 2 and Python 3 the file naming convention is

*title*_ *python version*_r*revno*.py

where :-

- ***title*** = software name, see 6.2.1 on
- ***python version*** = 2 or 3
- ***revno*** = integer number starting a '1' to differentiate updates to sample s/w code

For example if you are looking for the basic clock python code described in 6.2.1 to run in Python version 2 the file name will be;

*basicclock_2_r1.py*

[Remember that file names are case sensitive in Linux]

## 6.2 Sample applications

The sample applications have been purposely written with the aim that they are as easy as possible for people with limited knowledge of Python to understand. Duplicate versions coded for Python 2.xx and 3.xx are available for down load. All the applications need to be run with 'Administrator' rights, for instance

*sudo python basicclock_2_r1.py*

This is applies to any s/w which needs to access or control the Pi's GPIO lines.

It is recommended that they are run either from the console or in a terminal window. The applications can also be run through 'IDLE' but to do this you need to run the IDLE session with Administrator rights. To do this open a new terminal window and enter the instructions 'sudo IDLE' or sudo IDLE3.

Depending on which Pi hardware you are using and what else you have running you will see occasional display flicker, this is normal. It happens when the Linux operating system 'grabs' enough time [ for other processes ] to delay the dynamic update of the display to the point it is noticeable to the human eye. This most noticeable with the A+ and least noticeable with the B model 2. Running the application from the local console generally provides the best performance as this has the least overhead in terms of other 'CPU time hungry' processes running on the Pi.

The sample applications were developed and tested on Python 2 version 2.7.3 and Python 3 version 3.2.3 .

### 6.2.1 basicclock

'*basicclock*' is a simple single purpose clock demonstration application, it will run on either the basic or enhanced versions of the PiMuxClock. It will not respond to temperature 'push' in any way.

There are a number of options that are available in the code.

> **Time format**, this can be set to either 12 or 24 hour mode by commenting out the relevant lines at the start of the 'main mux display loop'.
>
> **Multiplex timings,** this should be fine as set but on slower Pi's adjusting these values may help improve display performance slightly. Setting '*muxtime*' to a much longer delay [ the default value is 0.005 ] is a good way to show how the multiplex process works. The '*muxtime*' definition can be found in the 'set up display variables' section of code.
>
> *File name;*

| | |
|---|---|
| *basicclock_2_r1.py* | - python 2.xx version |
| *basicclock_3_r1.py* | - python 3.xx version |

### 6.2.2 enhancedclock

'*enhancedclock*' extends the basic functionality of '*basicclock*' to include a room temperature display function. To use the temperature display function you need to have enabled the 1-wire interface on GPIO4 [see 5.3.1 'Configuring the 1 Wire interface'].

In normal mode the time is displayed, to see the room temperature hold down the temperature push [S1] until the display blanks then release, after a short delay the temperature will be display for around 2 seconds then the display will revert automatically to the time display.

Note:- as coded the '*enhancedclock*' will not correctly display temperatures below zero Celsius.

There are a number of options that are available in the code.

> **Time format**, this can be set to either 12 or 24 hour mode by commenting out the relevant lines at the start of the 'main mux display loop'.
>
> **Temperature format,** this can be set to either Centigrade [default] or Fahrenheit scales by uncommenting the indicated lines in the 'check for temp key' section of code.
>
> **Multiplex timings,** this should be fine as set but on slower Pi's adjusting these values may help improve display performance slightly. Setting '*muxtime*' to a much longer delay [the default value is 0.005] is a good way to show how the multiplex process works. The '*muxtime*' definition can be found in the 'set up display variables' section of code.
>
> *File name;*

| | |
|---|---|
| *enhancedclock_2_r1.py* | - python 2.xx version |
| *enhancedclock_3_r1.py* | - python 3.xx version |

### 6.2.3 muxprotemp

'*muxprotemp*' is a simple single purpose application that displays the Pi's CPU temp, it will run on either the basic or enhanced versions of the PiMuxClock. The temperature 'push' S1 and DS18B20 are <u>not</u> used by this application.

There are a number of options that are available in the code.

**Temperature format,** this can be set to either Centigrade [default] or Fahrenheit scales by uncommenting the indicated lines in the 'main mux display' section of code.

**Multiplex timings,** this should be fine as set but on slower Pi's adjusting these values may help improve display performance slightly. Setting '*muxtime*' to a much longer delay [ the default value is 0.005 ] is a good way to show how the multiplex process works. The '*muxtime*' definition can be found in the 'set up display variables' section of code.

*File name;*

| | |
|---|---|
| *muxprotemp_2_r1.py* | - python 2.xx version |
| *muxprotemp_3_r1.py* | - python 3.xx version |

### 6.2.4 seq_test

'*seq_test*' is a simple test program that can be used to check all the display digits are working correctly. There no options available in the code. The simply cycles through all the display combinations for each digit.

*File name;*

| | |
|---|---|
| *seq_test_2_r1.py* | - python 2.xx version |
| *seq_test_3_r1.py* | - python 3.xx version |

## 6.3 Your own Python coding

If you want to develop your own code for the PiMuxClock you can either start with one of the sample application or if you just want the hardware definitions you will find the file '*pimuxclock_port_ass*' which you can copy and paste into your application to save a bit of typing.

As part of the development testing the worst drive situation [all segments and digits driven] has been tested extensively. It is however not recommended that you leave the display operating in a static condition for extended periods as is could reduce the life time of the LED display.

# 7 Using your PiMuxClock

The best way to understand what you can and cannot do with your PiMuxClock is to try out the sample applications and go on from there, these are described in section 6.

The important thing to keep in mind is that there is no buffering so any software loops or extra functionality needs to be quick enough to update to next display digit faster than the update rate of the human eye ! In practice taking account of the time it takes to execute the actual 'GPIO.output' instruction you have about 5mS to play with between digits.

If you want to use the switch for any other purpose it is wired to GPIO line 23 and is active low – i,e. the IO line is driven low when you push the switch. *Do make sure this line is always configured as an input otherwise you risk damaging you Pi as the switch effectively shorts the line to ground*.

## 7.1 Connecting to the additional GPIO / RTC connections

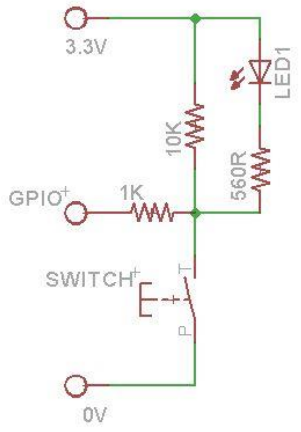


*Figure 4 'safe' GPIO connection circuit*

The 4 lines available on rear GPIO connector are wired directly to the main PI GPIO port. It is recommended you use the circuit [or similar] to that shown in Figure 4 to reduce the risk of damaging your Pi though accidental shorts or software errors.

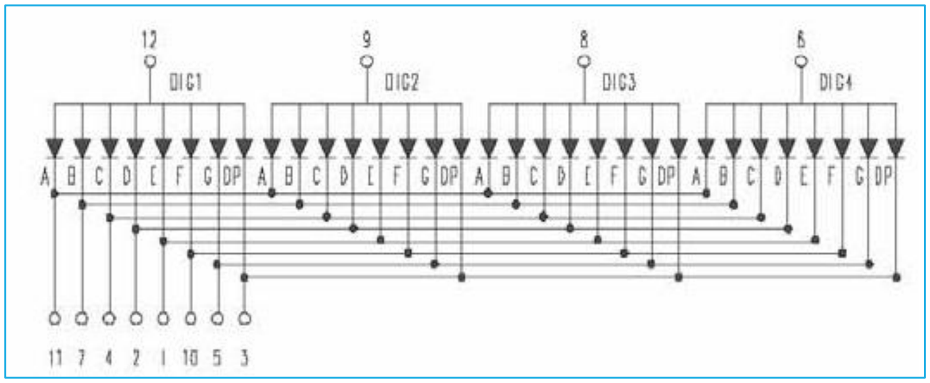
It is also possible to use pins 3 & 4 of the RTC connector as general IO if you do not want to fit the RTC. **Warning:-** the RTC connector has a 5 volt supply on pin 2 of the connector, connecting this directly to the PI's GPIO will damage your Pi.

# 8 Theory of Operation

## 8.1 Multiplexing basics

By commoning up [multiplexing] connections from displays the number of IO lines needed to drive multiple digits can be greatly reduced.

Figure 6 below shows how this is implemented for a 4 digit display.



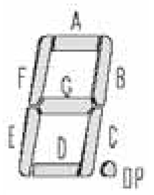


*Figure 5*
*7 segment display*

*Figure 6 Common Anode LED display- as used on the PiMuxClock*

In this example the ‘anode’ connections are all commoned. To display the desired character the cathode connections for the required segments are driven low, while the remaining segments are driven high together with desired character anode.

So for example if we wanted to display the number ‘one’ [i.e. segments B & C ] on digit 3;

Pins 7,4 together with 12,9,6 *would all be driven low*

With

Pins 11,2,1,10,5,3 together with 8 *would all be driven high*

By repeating this for each digit in quick succession our brain is fooled into seeing the display as though each character is continuous active.

# 9 Project Extension ideas

In this section some ideas for extension projects are outlined, the intent is to get you thinking about other things you can do with a PiMuxClock.

**Reaction timer game**, the idea here is to test the player's reaction time by seeing how long they tack to react. To build this you could use the switch on the board or wire another to one of the spare GPIO connections - see 11.2 .

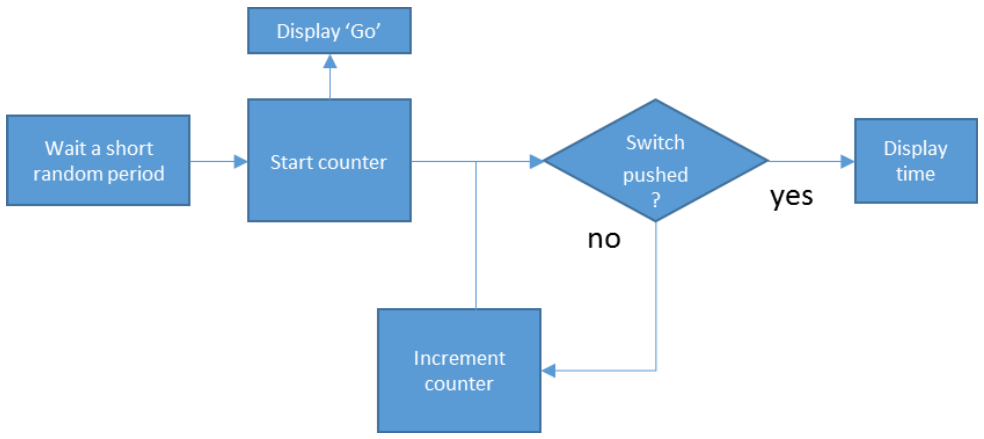
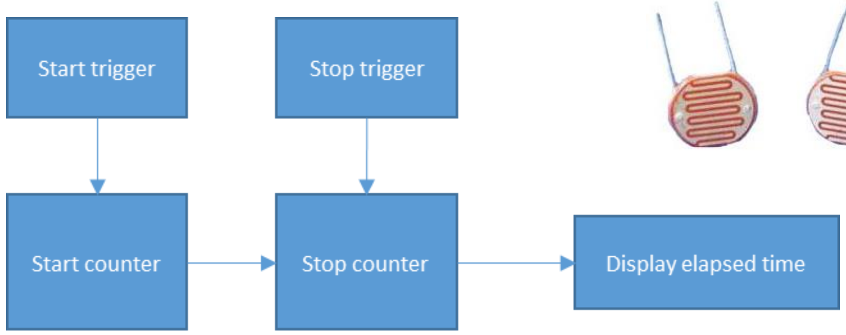


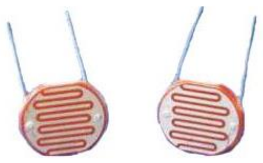
*Figure 7 Reaction timer*

**Adding an alarm**, it is very easy to add a buzzer to your PiMuxClock using one of the spare GPIO connections. From here you can turn a PiMuxClock into your own Pi based alarm clock. One thing to consider with selecting a buzzer is the power requirement;

- Is it low enough power to run directly from the Pi [ ie about 2-3mA @ 3.3V ] ?
- Is it 'externally driven', these require you to generate a waveform at a specific frequency to cause them to resonate?

**Elapsed time**, this could be used as a timer for another experiment. Here you are looking to use 2 of the spare GPIOs to start and stop a software timer, rather than use a mechanical switch you could look to use light sensitive resistors to detect a broken light beam.





**Display brightness adjustment**, the brightness level can be adjusted by modifying the time each display character is active  - essentially creating a simple from of pulse width modulation [PWM] . To do this you would need modified character display routine.

**Threading,** to keep things simple the example python programs supplied work using a simple timing loop to control the multiplexing. This however does make it difficult to do much else in the program. One solution to this is the use pythons 'threading module' this allows multiple processes to run at the same time within a single program. Here you might setup the display multiplex code as a separate thread which periodically checks for an updated variable to display from the main program loop. Threading can be quite complex to get working not least because many of the examples on-line are overly complicated, but is it powerful tool once mastered!

# 10 Problems

The PiMuxClock design is intentionally simple so hopefully you should not have any problems, the following list is based experience gained during developing testing the board and s/w.

**Pi will not start with PiMuxClock connected**

- *Check you have connected the PiMuxClock correctly and fully seated into the Pi's GPIO connector*

**You get a runtime error when you try to run any of the applications**

- *Make sure you run the application with root privileges - i.e. 'sudo python basc…..'*

**PiMuxClock does work at all**

- *Built from kit, check against section 3 to be sure that you have soldered everything in the correct place*
- *Check your pi by connecting an LED to one of the GPIO used by the PiMuxClock and see if it comes on when you run one of the applications*

**The display characters are not displaying correctly**

- *Check* for solder bridges causing a short circuit, the most likely place is on the 40 way GPIO *connector*

**One digit does not work or one segment on all the digits does not work**

- *Check you have connected the PiMuxClock correctly and fully seated into the Pi's GPIO connector*
- *Using* the table in the appendix [11.1] work out which line is the issue and look for dry joint in the circuit. For instance if digit 3 does not work this is driven from GPIO line 22 which is physical pin 15. With this *information* you can trace the circuit through to the display pin.

**The temperature function does not work**

- Check the *DS18B20* is soldered in correctly
- Re-check the configuration instructions listed in section 5.3.1

**The I2C [ RTC ] and / or temperature function stops working after updating the pi software [eg by performing an '*apt-get upgrade*' command]**

- Check the Pi forums to see if other people have seen the same problem with I2C or 1-wire interfaces, it is possible there has been a change to way these interfaces need to be setup

# 11 Appendix, Technical Spec Information

## 11.1 GPIO assignments

Important:- All references to GPIO line numbers and the sample s/w in this document refer to BCM numbering.

| Function Definition | | GPIO no. [BCM] | GPIO no. [physical] | |
|---|---|---|---|---|
| **Segments** | | | | |
| | segA | 5 | 29 | |
| | segB | 6 | 31 | |
| | segC | 13 | 33 | |
| | segD | 19 | 35 | |
| | segE | 26 | 37 | |
| | segF | 12 | 32 | |
| | segG | 16 | 36 | |
| | segDP | 20 | 38 | |
| **Digit** | | | | Left to right when viewed display from front |
| | char1 | 17 | 11 | |
| | char2 | 27 | 13 | |
| | char3 | 22 | 15 | |
| | char4 | 18 | 12 | |
| **Extended GPIO setup** | | | | |
| **Temperature key** | | | | |
| | tempkey | 23 | 16 | |
| **Extra GPIO connections** | | | | |
| | gp24 | 24 | 18 | |
| | gp25 | 25 | 22 | |
| | gp14 | 14 | 8 | Auto configured as UART at power up [TXD] |
| | gp15 | 15 | 10 | Auto configured as UART at power up [RXD] |
| **DS1307 I2C interface** | | | | |
| | SDA | 2 | 3 | I2C Data |
| | SDL | 3 | 5 | I2C Clock |
| **DS18B20 1-wire interface** | | | | |
| | data | 4 | 7 | Data connection to DS18B20 |

*Table 2 GPIO assignments*

## 11.2 I2C and additional GPIO connections

Figure 8 details the I2C and additional GPIO connections as viewed from the back of the PiMuxClock board.

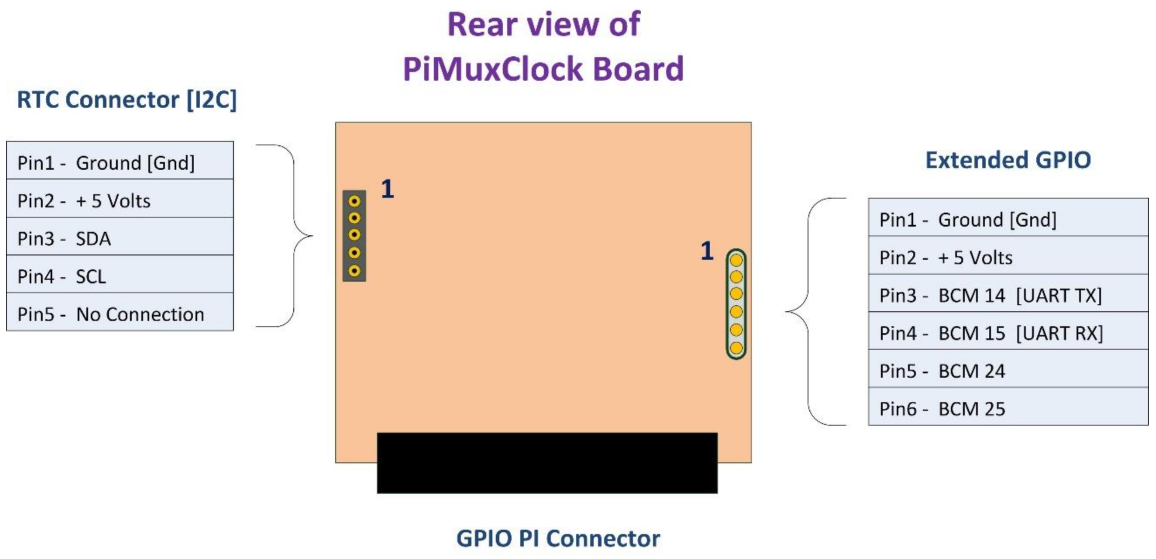


*Figure 8 Review view of PiMuxClock board*

## 11.3 Parts List

Parts are available from a number of suppliers, although you may need to shop around for the 40 way right-angle connector and 620 Ohm resistors. All of the parts can be purchased from Bitsbox in the UK who have helpfully added the right angle connector to their stock range.

| Ref | Part Description | Quantity |
|---|---|---|
| RPI GPIO Conn | 40W [20+20] right angle connector | 1 |
| R1,2,4,5,6,7,8,9 | 620 Ohm 1/4W | 8 |
| R10 | 10K Ohm 1/4W | 1 |
| R3 | 4k7 Ohm 1/4W | 1 |
| Display | 4 digit 0.56 common anode [ 12 pin variant ] | 1 |
| Temp Sensor | DS18B20 | 1 |
| Tactile Switch | Black Tactile 7mm switch [spst] | 1 |
| RTC Conn | 5-Way Single Row Socket | 1 |
| GPIO exp Conn | 6 -Way Header Strip | 1 |

*Table 3, part list*

Points to watch if you are trying to purchase replacement parts;

- *The display needs to be common anode type, with 12 pins [ see Figure 6 for correct pin out ]*
- *The display needs to be 0.56" segment size*
- *Make certain you are buying a right angle [$90^{O}$] 40 pin connector*
- *620 Ohm resistors are not that common, 680 Ohm are fine but the display will be slightly dimmer*