The Wombat

Prototyping Board for Raspberry Pi

Operation and Software Guide

This prototyping board is intended to make it easy to experiment and try out ideas for building electronic devices that connect to the Raspberry Pi.

It works with the Raspberry Pi models which include a 40-pin GPIO header, such as the A+, B+, and the Raspberry Pi 2 model B[1].

The Wombat board breaks those 40 GPIO pins out to a labelled single in line (SIL) header alongside a solderless breadboard, and adds a number of useful facilities including analog inputs, a 500 mA 3.3V power supply, a USB serial interface, and pushbutton switches, LEDs and a potentiometer for testing.

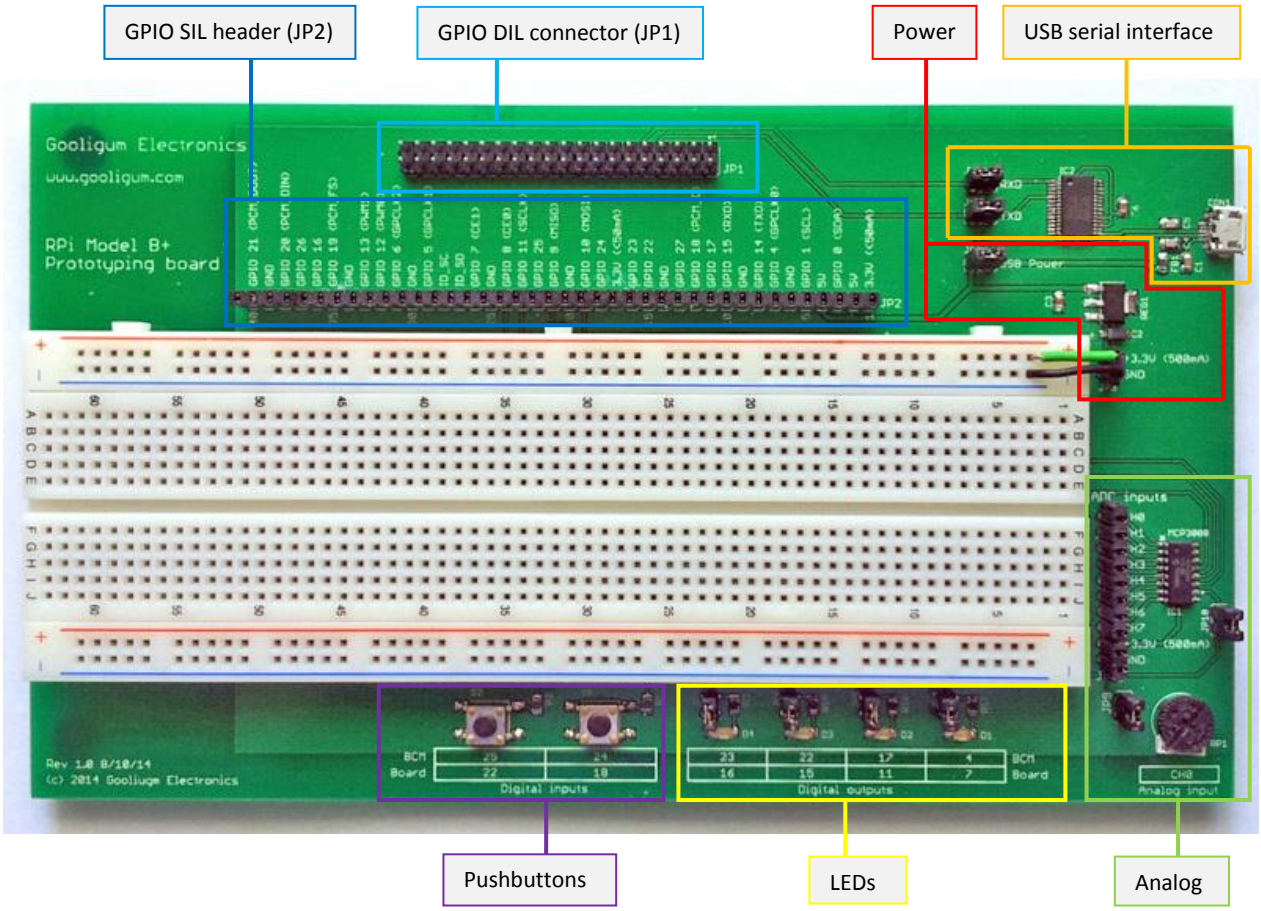
It features:

- All 40 Raspberry Pi (2, 3, A+ or B+) GPIO pins broken out to a clearly labelled and easily accessible 40-pin header alongside a large solderless breadboard
- Onboard 500 mA 3.3 V voltage regulator
- 8 x 10-bit analog inputs, via an analog-to-digital converter (MCP3008) connected to the Raspberry Pi's SPI bus
- 2 x pushbutton switches to use as digital inputs
- 4 x LEDs to use as digital outputs
- 1 x potentiometer to use as a basic analog input
- FTDI USB to serial interface, making the Raspberry Pi's serial port accessible via a Micro USB connector
- Ability to power the Raspberry Pi via the Wombat board's Micro USB connector (meaning that a single Micro USB cable can provide both power and serial connectivity)
- USB power and all onboard devices can be disconnected or disabled via jumpers

[1] The Wombat board can also be used with the original Raspberry Pi models A and B, which have a 26-pin GPIO header, via an appropriate 26 to 40 pin cable (not supplied) which connects the Raspberry Pi's 26 GPIO pins to pins 1-26 on header JP1.

Hardware Operation

The Wombat board consists of a number of functional blocks, as shown[2]:



Setup

Shutdown and remove power from your Raspberry Pi, and then use the supplied 40-pin IDC cable to connect the Raspberry Pi's GPIO port (J8) to the Wombat board's GPIO connector (JP1).

Ensure that pin 1 of each port is plugged in to pin 1 of the cable (indicated by an arrow on each socket, and by a differently coloured wire along the "pin 1" edge), and the that the IDC sockets are properly aligned with both rows of pins.

You must now decide how to power your Raspberry Pi:

- If you choose to power the Raspberry Pi in the traditional way, via the Raspberry Pi's Micro USB power connector, **remove jumper JP13** (marked 'USB Power').

  This will ensure that, if you later use the Wombat board's USB interface as a serial port, there will be no interference between the Raspberry Pi's power supply and the Wombat board's USB power.

[2] your board may differ slightly from the "rev 1.0" board used for this photo, but the various features, connectors and jumpers are all in the same place and operate the same way.
Note also that the jumper wires shown plugged into the 3.3 V power header (JP3) are not supplied with the board, but demonstrate how 3.3V power would normally be connected to the breadboard

- If you choose to power the Raspberry Pi via the Wombat board's Micro USB connector[3] ('CON1'), close jumper JP13 ('USB Power') and **do not apply power to the Raspberry Pi's Micro USB power connector**. Then connect a 5 V supply with a minimum rating of 1.3 A (but 2 A recommended) to the Wombat board's Micro USB connector.

  Note that some laptop USB ports, designed to act as fast charging ports, are suitable for powering the Raspberry Pi via the Wombat board (while also providing a USB serial connection, if desired), but standard USB 2.0 ports, which are only designed to supply up to 500 mA, are likely to be inadequate for powering the Raspberry Pi.

CAUTION: DO NOT apply power to both the Raspberry Pi's Micro USB power connector and the Wombat board's Micro USB connector at the same time if JP13 ('USB Power') is closed

You can now power your Raspberry Pi via either Micro USB connector (but not both at the same time!) and use it as normal.

GPIO header

All of the 40 GPIO pins are brought out to the 40-pin SIL header, JP2.

The pin numbers along the bottom of the header (next to the breadboard) correspond to the Raspberry Pi's GPIO port pin numbers – if you program in Python and use the RPi.GPIO module, these are the numbers used in the "BOARD" pin numbering mode.

The labels along the top of the header correspond to the Broadcom documentation and are more commonly used – these are the numbers used in the RPi.GPIO module's "BCM" pin numbering mode. The labels also show the most commonly used alternate function for each pin.

For example, pin 10 is marked '10' on one side and 'GPIO 15 (RXD)' on the other. It would be accessed in BOARD mode as pin 10, or in BCM mode as pin 15. This pin is also used as the RXD (receive serial data) pin.

Power supply

The Raspberry Pi's GPIO connector supplies 5 V power on pins 2 and 4, with effectively as much current available as the external power supply is able to provide.

The Raspberry Pi's GPIO connector also supplies 3.3 V on pins 1 and 17, but this is limited to 50 mA in total across those pins. Given that the GPIO pins operate at 3.3 V, this 50 mA 3.3 V power supplied by the Raspberry Pi may be inadequate for your projects, which is why the Wombat board includes a 3.3 V regulator (REG1) able to supply up to 500 mA.

This 500 mA 3.3 V supply is made available on header JP3 and at the bottom of header JP8[4], aligned with the power rails on the breadboard.

It is recommended that you only use these 500 mA 3.3 V power connections on JP3 and JP8, and **ignore the 50 mA 3.3 V power pins on GPIO header JP2**.

CAUTION: DO NOT connect the Wombat board's 500 mA 3.3 V power supply to the Raspberry Pi's 3.3 V power rail via the GPIO header. Keep them separate.

---

[3] powering the Raspberry Pi via its GPIO header pins in this way is often referred to as "back powering"

[4] the 500 mA maximum refers to the total current supplied by these two headers

As mentioned earlier under "setup", the Raspberry Pi's 5 V power rail (available via GPIO pins 2 and 4) can optionally be connected to the Wombat board's Micro USB connector by closing jumper JP13, labelled 'USB Power'.

Note that the 3.3 V regulator is powered from the Raspberry Pi's 5 V power rail, regardless of whether the 5 V power is supplied from the Raspberry Pi's onboard Micro USB power connector or the Wombat board's Micro USB connector – **jumper JP13 has no effect on the Wombat board's 500 mA 3.3 V power supply**, which is always available as long as the Raspberry Pi is powered.

USB serial interface

If you connect the Wombat's Micro USB port to any standard USB port on your laptop or desktop PC[5], your PC should see the board as a virtual COM port (VCP).

You may however need to install the FTDI VCP driver first.

These drivers are available for current versions of Windows (going back to XP), Mac OS, and most Linux distributions, and are available from: http://www.ftdichip.com/Drivers/VCP.htm

However, you are likely to find that the FTDI VCP driver is already installed, or, as with recent versions of Windows, will be installed for you automatically.

Once the FTDI driver is installed and the Wombat board connected, you will need to determine which virtual serial ("tty" or "COM") port the board is visible as. In Windows, you can look in the Device Manager for "USB Serial Port" – if you have more than one of these, try unplugging the Wombat board to see which USB Serial Port device disappears. It will appear as something like "COM3" under Windows, or perhaps "/dev/ttyUSB0" if you are using Linux.

Assuming that you have not disabled your Raspberry Pi's serial console, or modified its configuration (if you don't know how, or didn't know that you could, you probably haven't...), and after ensuring that jumpers JP11 (RXD) and JP12 (TXD) are closed, you can now use your favourite terminal program (such as PuTTY for Windows) to connect to this COM port, with it set to 115200 bps.

You many need to press "Enter", but you should then be greeted by the familiar Raspbian login prompt and you can login to your Raspberry Pi via the USB serial interface.

If you are using Raspberry Pi's serial receive (GPIO 15 / RXD) or transmit (GPIO 14 / TXD) pins for something else[6] and do not want to have the FTDI chip connected to either of those pins, simply remove jumper JP11 (RXD) and/or JP12 (TXD) to disconnect the FTDI chip from those pins.

LEDs

The four LEDs, D1 – D4, are connected, via jumpers and 220 Ω resistors, to GPIO pins as marked on the board.

Labels are provided for both the "BCM" and "BOARD" pin numbering schemes.

For example, LED D1 would be accessed in BCM mode as pin 4, or in BOARD mode as pin 7.

To light an LED, set the corresponding GPIO pin to "high".

If you are using the pin for something else and do not want the LED to light when that pin goes high, simply remove its associated jumper. For example, to disable D1, remove JP4.

---

[5] you will need a standard "Micro USB" cable (not supplied), as commonly used with Android mobile phones

[6] maybe you're building a device that connects to the Raspberry Pi via the serial port, or maybe you simply need to use a lot of GPIO pins in your project

Note: LED D1 is connected to the GPIO 4 pin which, when the Raspberry Pi starts up, is configured by default as an input, with an internal pull-up resistor enabled. That pull-up resistor supplies a small current which can be enough to dimly light the LED. So if you see that D1 is dimly lit, don't worry – the Wombat board and your Raspberry Pi are working correctly and, if your program uses GPIO 4 as an output, D1 will turn on and off as described above.

Pushbutton switches

The two pushbutton switches, S1 and S2, are connected via 1 kΩ isolation resistors to GPIO pins as marked on the board.

Labels are provided for both the "BCM" and "BOARD" pin numbering schemes.

For example, switch S1 would be accessed in BCM mode as pin 24, or in BOARD mode as pin 18.

The pushbuttons are *active low* – pressing the switch connects the GPIO pin to ground, pulling the input low. Therefore, to use one of these switches as a digital input, you should enable the internal pull-up resistor for that pin.

For example, using the RPi.GPIO module with Python, to setup switch S1 as an input and LED D1 as an output:

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)          # use BCM pin numbering

GPIO.setup(4, GPIO.OUT)                                 # LED D1 on GPIO4
GPIO.setup(24, GPIO.IN, pull_up_down=GPIO.PUD_UP)   # button S1 on GPIO24
```

and then to light the LED whenever the button is pressed, you could use:

```
while True:
    if GPIO.input(24) == 0:             # if button is pressed (low)
        GPIO.output(4, 1)               #   light the LED
    else:                               # else
        GPIO.output(4, 0)               #   turn it off
```

or, more succinctly:

```
while True:
    GPIO.output(4, not(GPIO.input(24)))
```

However, if you use Python with the 'wombat.py' module supplied for use with the Wombat board (see the Software section, later), all of this code becomes simply:

```
from wombat import *

while True:
    setled(1, readbutton(1))
```

If you use the wombat.py module, you don't need to be aware of details such as enabling pull-ups on the switch inputs. But if you're not using Python, or don't want to use the supplied module, it's important to be aware of how these switches are connected.

Note that there are no jumpers associated with the pushbutton switches. These switches can only affect your circuit if the button is pressed, so if they are not being used they will have no effect on your circuit. And if an unused button is pressed accidentally, the 1 kΩ isolation resistor means that it won't damage the GPIO pin, even if it is configured as an output.

Analog inputs

The Wombat board includes a MCP3008 analog-to-digital converter (ADC) chip, which provides eight 10-bit analog inputs.

These inputs (or "channels") are made available on header JP8 and are labelled 'CH0' to 'CH7'.

The inputs are referenced to the Wombat board's 500 mA 3.3 V power supply – an input of 0 V reads as 0 while an input of 3.3 V will read as 1023 (= 2^{10}-1).

The MCP3008 chip is connected to the Raspberry Pi's SPI bus 0 as device 0 and is accessed as Linux device '/dev/spidev0.0'.

Note that, in the standard Raspbian Linux distribution (the operating system used on most Raspberry Pis), the Linux SPI driver is disabled by default, and must be enabled before the Wombat board's analog inputs can be used. To enable the SPI driver, run the 'raspi-config' utility:

```
sudo raspi-config
```

then select "Advanced Options" and then "SPI", answering "Yes" to both "Would you like the SPI interface to be enabled?" and "Would you like the SPI kernel module to be loaded default?".

After you exit the 'raspi-config' utility and then reboot your Raspberry Pi, the SPI driver should be enabled and the '/dev/spidev0.0' device file should exist.

Your program can use the SPI bus to send the appropriate commands to and read data from the MCP3008; see the MCP3008 data sheet available from www.microchip.com if you are interested in all the low-level details.

However, if you use Python with the supplied 'wombat.py' module, you can use the "`readadc()`" function it provides, without having to be aware of these details.

For example, to print the value of analog input CH0 every 0.5 seconds:

```
import time
from wombat import readadc

adc_chan = 0        # use analog input CH0

while True:
    # get current value (0-1023) of ADC input
    adc_out = readadc(adc_chan)

    sensor_voltage = (adc_out * 3.3)/1024
    print "input voltage = %5.3f V" % (sensor_voltage)
    time.sleep(0.5)
```

See the Software section later, for details.

If you don't need the analog inputs and want to use some other device as device 0 on the SPI bus, remove jumper JP10. This will disable the MCP3008 chip, preventing it from interfering with other devices connected to the Raspberry Pi's SPI pins.

A 10 kΩ potentiometer ('RP1') is available as an analog voltage source (0 – 3.3 V) on input CH0.

You can use it to test your program (test that your code responds correctly to various input levels, before using it with an analog sensor), or simply as an analog input.

But if you want to use input CH0 for something else (perhaps your project needs to use all eight analog inputs), you can disconnect RP1 from CH0 by removing jumper JP9.

Jumper summary

As described above, jumpers are used to enable each function, bringing different elements in or out of the circuit.

Here is a summary of what each jumper is used for:

Jumper	Function
JP4	Connects LED D1 to pin GPIO 4
JP5	Connects LED D2 to pin GPIO 17
JP6	Connects LED D3 to pin GPIO 22
JP7	Connects LED D4 to pin GPIO 23
JP9	Connects trimpot RP1 to analog input CH0
JP10	Connects MCP3008 IC to SPI bus chip select pin GPIO 8 (CE0) Removing this jumper holds the MCP3008's chip select pin high, placing the device in low-power standby mode.
JP11	Connects USB-serial (FT232RL) TXD pin to serial port pin GPIO 15 (RXD)
JP12	Connects USB-serial (FT232RL) RXD pin to serial port pin GPIO 14 (TXD)
JP13	Connects Micro USB connector (CON1) VBUS (+5 V) pin to Raspberry Pi's internal 5 V rail (GPIO header pins 2 and 4). Remove this jumper if you are using the Raspberry Pi's onboard Micro USB power connector.

Software

A Python module, "wombat.py", has been written specifically to make it easy to use the Wombat board's features, including the analog inputs, in Python programs[7].

Installation

These instructions assume that you are using the "Raspbian" Linux distribution (it's the most commonly used Raspberry Pi operating system) and that you're using the command line interface and your Raspberry Pi is networked and is able to access the Internet.

Before installing the wombat.py module, you should first ensure that your Raspberry Pi is up to date, with:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo reboot
```

You should then enable the SPI interface on your Raspberry Pi, by running the 'raspi-config' utility:

```
sudo raspi-config
```

then select "Advanced Options" and then "SPI", answering "Yes" to both "Would you like the SPI interface to be enabled?" and "Would you like the SPI kernel module to be loaded default?".

After you exit the 'raspi-config' utility you should again reboot your Raspberry Pi:

```
sudo reboot
```

To check that the SPI interface has been enabled and the kernel module loaded, run 'lsmod':

```
lsmod
```

You should see the entry "spi_bcm2708" or "spi_bcm2835" listed.

We then need to install the python-dev modules:

```
sudo apt-get install python-dev -y
sudo apt-get install python3-dev -y
```

And then install the py-spidev module[8]:

```
sudo apt-get install git -y
git clone git://github.com/doceme/py-spidev
cd py-spidev
sudo python2 setup.py install
sudo python3 setup.py install
cd ..
```

[7] the module supports both Python 2 and Python 3

[8] these commands may differ from some of the instructions you'll find online, but they work on the current versions of both the "Wheezy" and "Jessie" Raspbian distributions (at the time of writing – March 2016) and install py-spidev for use with both Python 2 and Python 3

You can now install the Wombat code library:

```
git clone https://github.com/gooligumelec/wombat-code.git
```

This will create a "`wombat-code`" directory. It contains the "`wombat.py`" module, along with some demo programs which demonstrate how to use the various wombat functions.

To install the "`wombat.py`" module, navigate to the new directory:

```
cd wombat-code
```

and then type:

```
sudo python2 setup.py install
sudo python3 setup.py install
```

To use the wombat.py module, add this line toward the start of your Python program:

```
from wombat import *
```

You can then refer to the various functions (described below), such as readadc(), directly in your program. For example, to read analog input channel 2:

```
    adc_out = readadc(2)
```

Or, in the unlikely event that the name of any of the wombat.py functions clashes with another function (perhaps from another module that you have imported), you can instead use:

```
import wombat
```

and use the "wombat." prefix to refer to the functions, for example:

```
adc_out = wombat.readadc(2)
```

Each function provided by the wombat.py module is described below.

readadc()

This function is used to read the eight analog (ADC) inputs.

It returns the current value of a specific input channel as an integer between 0 and 1023, where 0 represents an input of 0 V and 1023 represents 3.3 V:

```
    value = readadc(chan)
```

where 'chan' is an integer between 0 and 7, corresponding to analog inputs CH0 to CH7.

For example, to print the value of analog input CH0 every 0.5 seconds:

```
import time
from wombat import readadc

adc_chan = 0        # use analog input CH0

while True:
    # get current value (0-1023) of ADC input
    adc_out = readadc(adc_chan)

    sensor_voltage = (adc_out * 3.3)/1024
    print "input voltage = %5.3f V" % (sensor_voltage)
    time.sleep(0.5)
```

setled()

This function is used to control the LEDs.

It sets the state of a specified LED – turning it on or off:

```
setled(n, state)
```

where 'n' is an integer between 1 and 4, corresponding to LEDs D1 to D4,
and 'state' is the output state: 0 (or False) turns the LED off and 1 (or True) turns the LED on.

For example, to flash D2 once per second:

```
import time
from wombat import *

while True:
    setled(2, 1)         # LED D2 on
    time.sleep(0.5)      #   for 0.5 sec

    setled(2, 0)         # LED D2 off
    time.sleep(0.5)      #   for 0.5 sec
```

readbutton()

This function is used to read the pushbutton switches.

It returns the state of a specified switch: 1 (or True) if the switch is pressed, 0 (or False) if not pressed:

```
value = readbutton(n)
```

where 'n' is an integer between 1 and 2, corresponding to switches S1 and S2.

For example, to light LED D1 whenever pushbutton S1 is pressed:

```
from wombat import *

while True:
    if readbutton(1):       # if button S1 is pressed
        setled(1, 1)        #   light D1
    else:                   # else it's not pressed, so
        setled(1, 0)        #   turn off D1
```

Note again that this program can be written more succinctly (if not as clearly) as:

```
from wombat import *

while True:
    setled(1, readbutton(1))
```

cleanup()

Finally, the cleanup() function simply resets all of the GPIO pins to their default configuration: output pins will cease being outputs and therefore any LEDs which are lit will turn off.

It doesn't take any parameters or return any values; simply place it at the end of your program to clean up any GPIO pins your program used:

```
cleanup()
```

For example, if we add "try" and "except" statements to our "light an LED when a button is pressed" program, we can clean up when the program is interrupted:

```
from wombat import *

try:
  while True:
    setled(1, readbutton(1))

except KeyboardInterrupt:
  cleanup()
```

Permissions (privileges) needed

If you're running the older "Wheezy" release of Raspbian, the settled(), readbutton(), readadc() and cleanup() functions all require "root" (or "superuser") privileges to run.

That means that, unless you are logged in as 'root'[9], you need to use the 'sudo' command to run any programs which use these functions.

For example, to run the supplied led_button-demo.py program, you would type:

```
sudo ./led+button-demo.py
```

If you're running the newer "Jessie" release, these functions no longer require special privileges and there is no need to use the 'sudo' command with Python programs using the Wombat support functions.

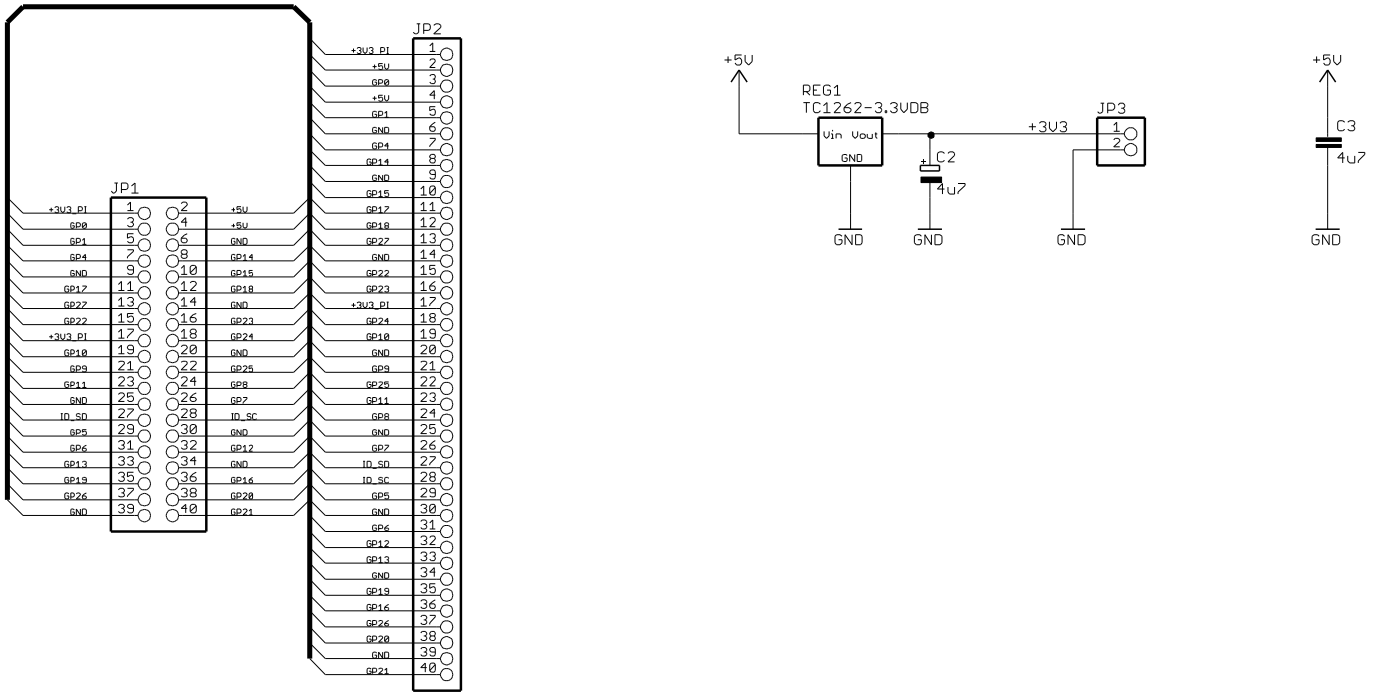
So, if you're using the "Jessie" release of Raspbian, to run the supplied led_button-demo.py program, you can simply type:

```
./led+button-demo.py
```

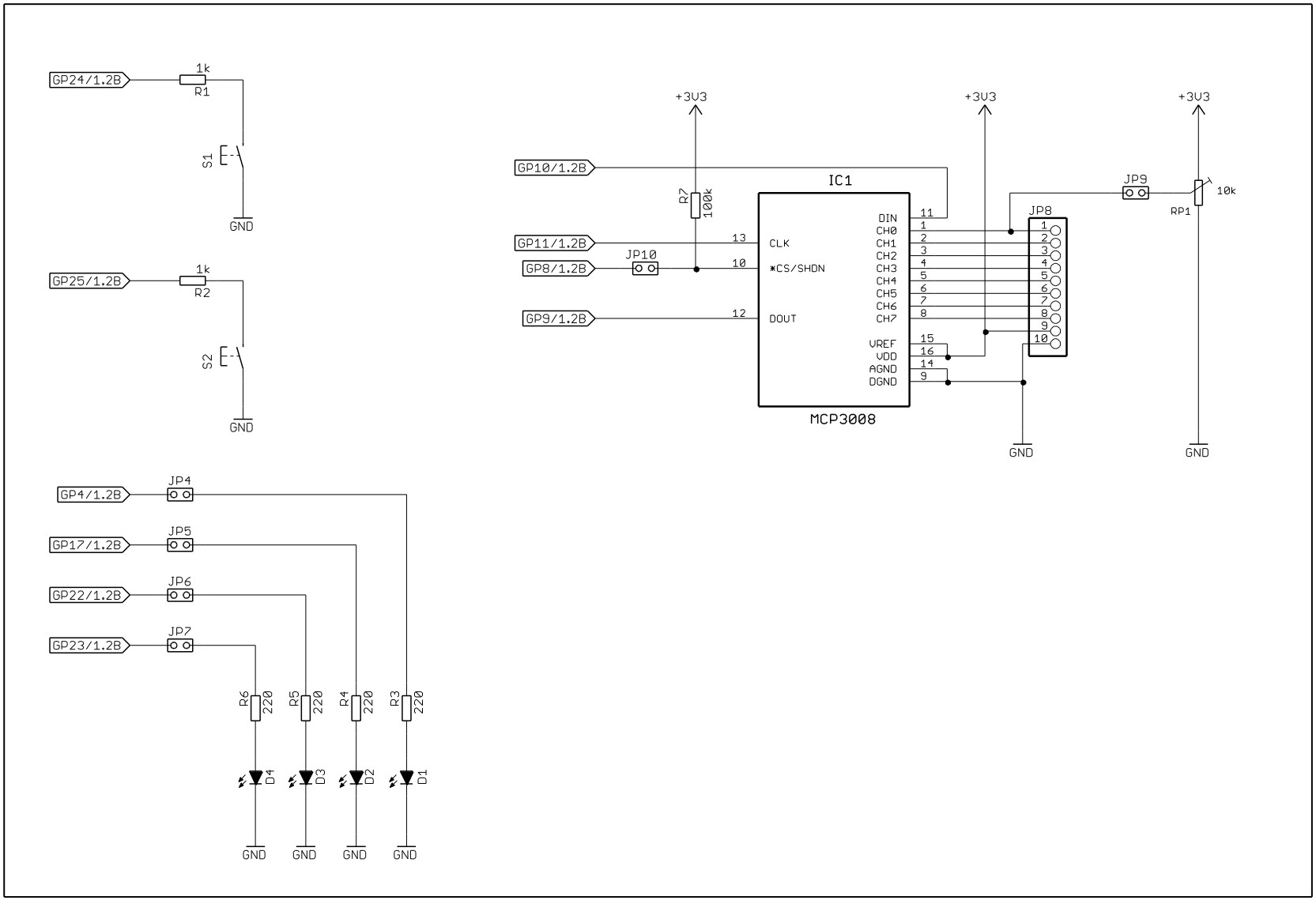
[9] logging in as 'root' may make your life easier but it's not recommended; it's too easy to break something by mistake...

Appendix A – Schematics

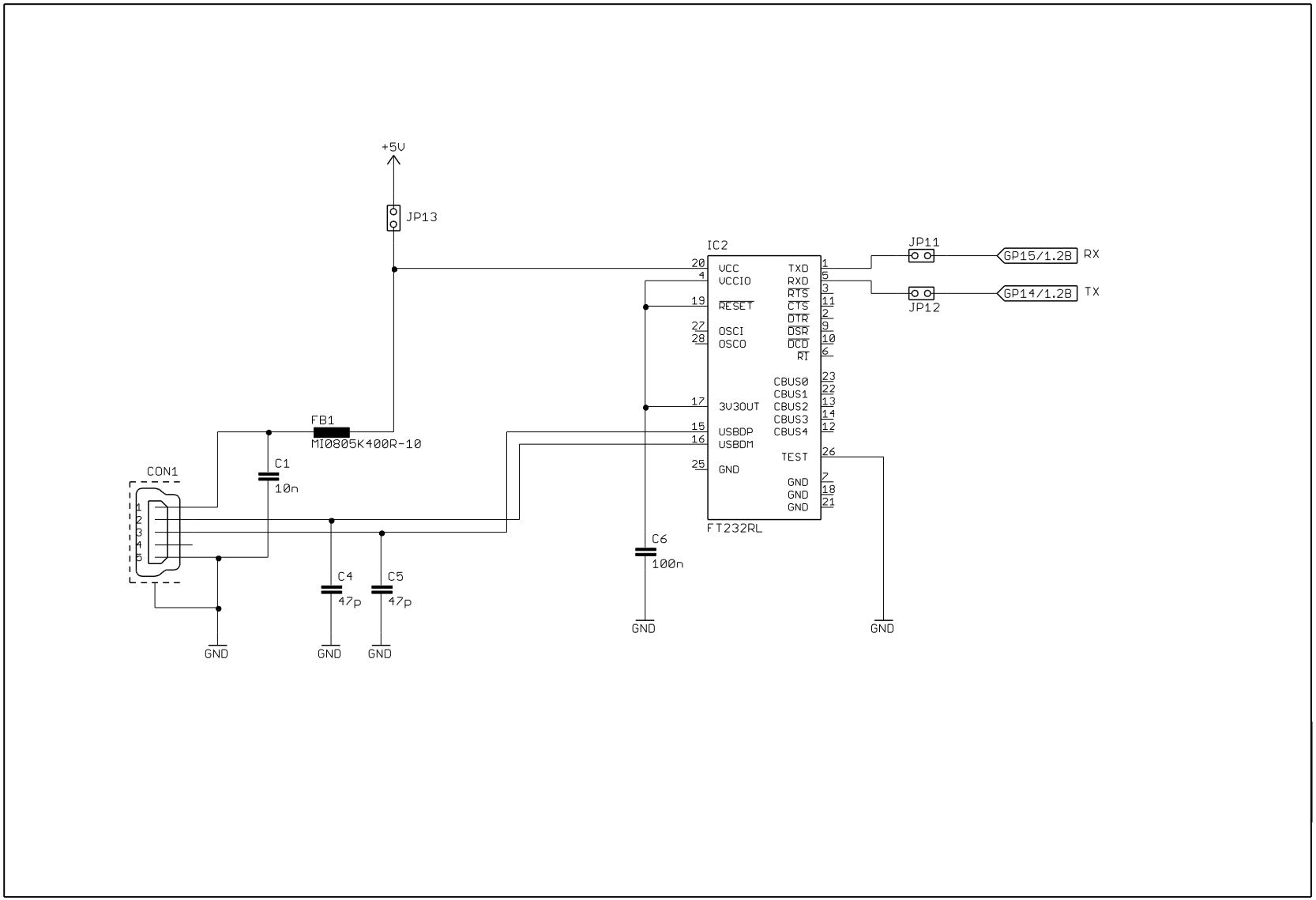
Sheet 1: Connectors + power



Sheet 2: Input/Output



Sheet 3: USB


+5V
JP13
IC2
VCC
VCCIO
RESET
OSCI
OSCO
3V3OUT
USBDP
USBDM
GND
TXD
RXD
RTS
CTS
DTR
DSR
DCD
RI
CBUS0
CBUS1
CBUS2
CBUS3
CBUS4
TEST
GND
GND
GND
FT232RL
JP11
JP12
GP15/1.2B
RX
GP14/1.2B
TX
FB1
MI0805K400R-10
C1
10n
CON1
C4
47p
C5
47p
C6
100n
GND
GND
GND
GND
GND