AuroraWatchNet magnetometer manual

For PoE and radio systems using Calunium microcontroller

Steve R. Marple,
Lancaster University.

2017-09-30 21:45:35 +0100
Commit: a5d1ac992f3c91ad536330f4207b4051dbea0f27

Licence

Contents

List of figures

Abbreviations

ADC	analogue to digital converter
API	application programming interface
DHCP	dynamic host configuration protocol
DIP	dual-inline package
DNS	domain name system
DTR	data terminal ready
EEPROM	electrically erasable programmable read-only memory
ESD	electro-static discharge
FAT	file allocation table
FET	field-effect transistor
GPU	graphics processing unit
HMAC	hash-based message authentication code
HTTP	hypertext transfer protocol
HTTPS	hypertext transfer protocol secure
I2C	inter-integrated circuit (bus)
IC	integrated circuit
IP	internet protocol
IR	infra-red
ISM	Industrical, scientific and medical (radio band)
ISP	in-circuit serial programmer (sometimes abbreviated as ICSP)
JTAG	joint test action group
LED	light emitting diode
MD5	message digest 5
NFS	network file system
NTP	network time protocol
PCB	printed circuit board
PoE	power over ethernet
RFI	radio-frequency interference
RTC	real-time clock
SD	secure digital
SOIC	small outline integrated circuit
SSH	secure shell
TCP	transmission control protocol
TTL	transistor-transistor logic
UDP	user datagram protocol
URL	uniform resource locator

USB	universal serial bus
UTC	coordinated universal time

Part I

Introduction

Chapter 1

Overview of the hardware

1.1 Introduction

The magnetometer is designed for low-power operation, simple installation and ease of construction. The entire design is open source, allowing anyone with reasonable soldering ability to construct one.

The magnetometer has two major parts, the base unit and the sensor unit (Figure 1.1). The sensor unit is located outdoors, away from buildings, cars and other sources of human disturbance. It is battery powered and communicates with the base unit by a radio link (433 MHz or 868 MHz), enabling the sensor to be installed without any wiring to the base unit. The base unit is placed indoors and should be positioned such that there are the minimum number of walls between it and the sensor unit.

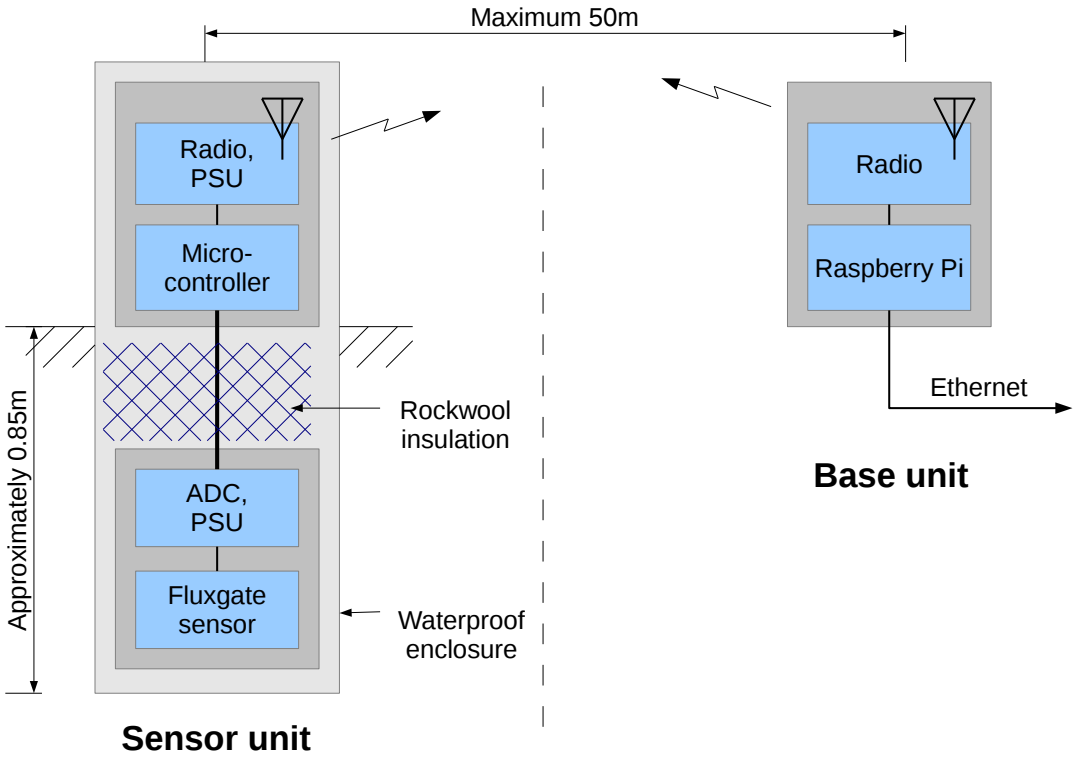


Figure 1.1: System overview.

1.2 Sensor unit

The sensor unit (figure 1.2) is contained inside a waterproof enclosure approximately 1.1 m high which is partially buried to reduce temperature variations and to provide a stable foundation. The sensor itself is placed at the bottom of the enclosure, approximately 0.85 m below ground. The microcontroller, radio module and battery are positioned in the top part of the enclosure, above ground level. Insulating material (e.g. rockwool) is used to fill the space in-between.

The Calunium microcontroller board is based on the popular Arduino platform but uses the more powerful Atmel ATmega1284P microcontroller.

Figure 1.2: Sensor unit.

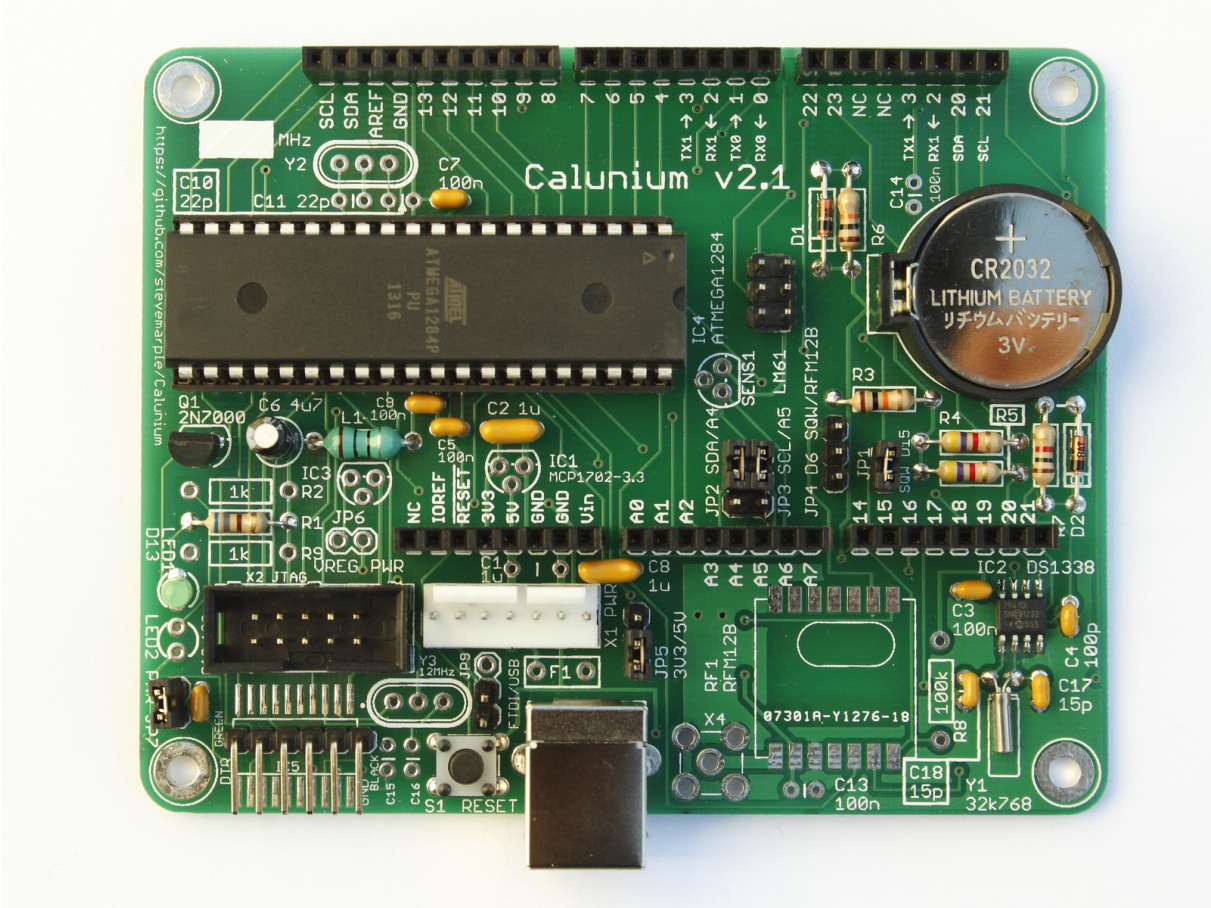


Figure 1.3: Calunium microcontroller PCB.

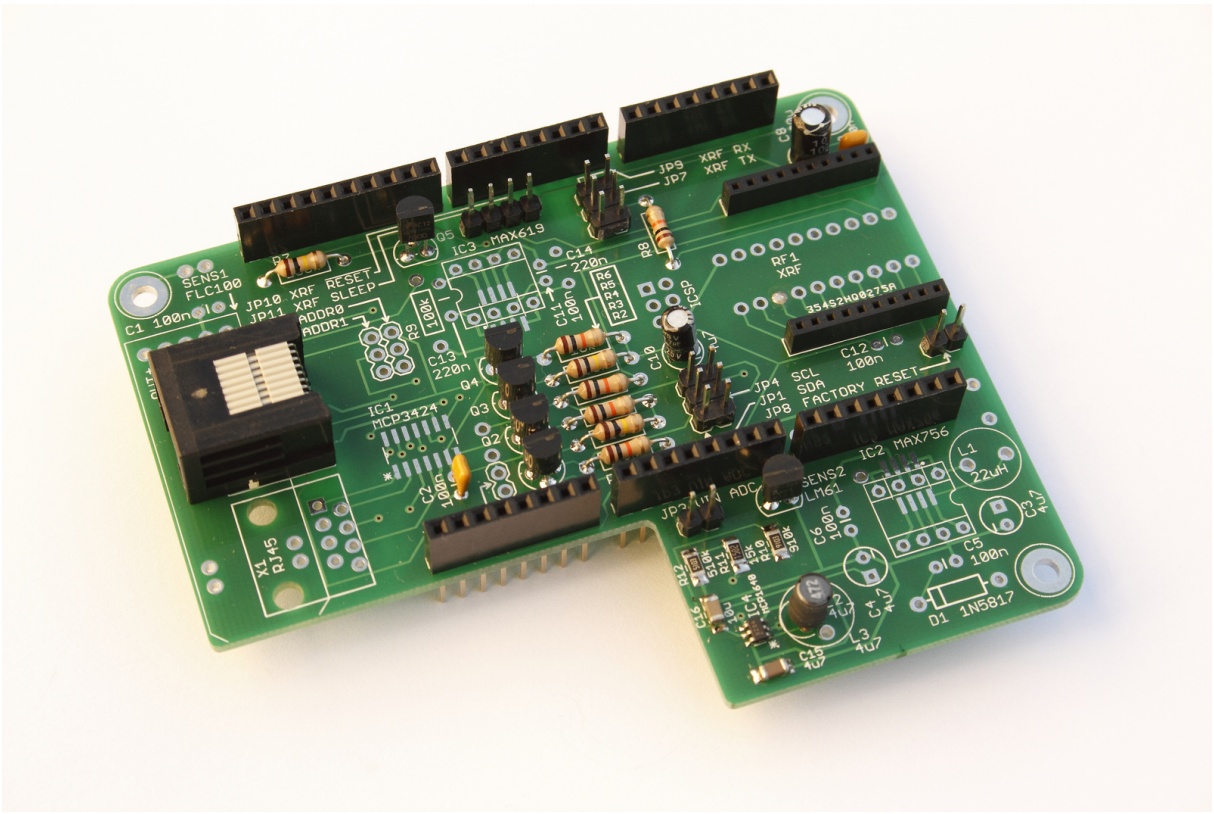


Figure 1.4: The FLC100 shield fits onto the Calunium microcontroller PCB.
© Steve Marple. CC BY-SA 2.0. http://www.flickr.com/photos/stevemarple/10787109594/

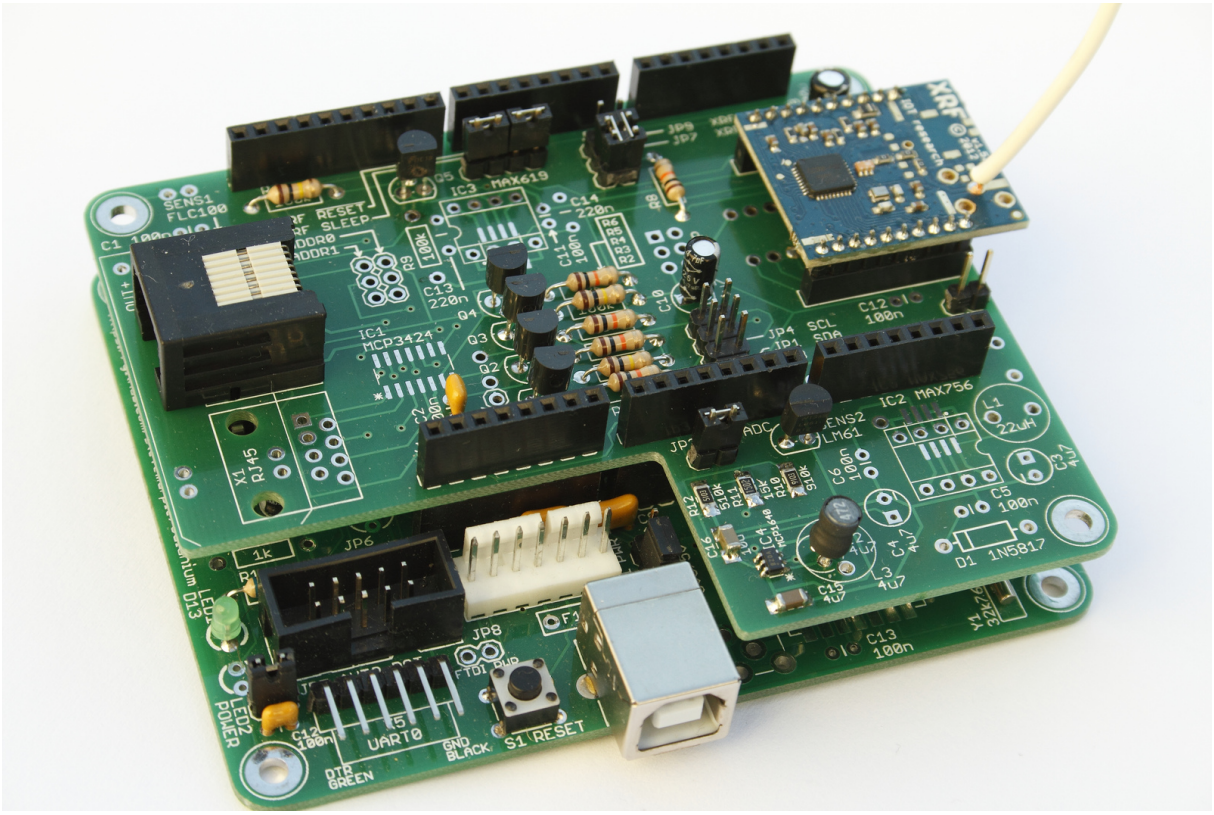


Figure 1.5: Calunium microcontroller board and FCL100 shield.
© Steve Marple. CC BY-SA 2.0. http://www.flickr.com/photos/stevemarple/10913562526/

1.3 Base unit

The base unit is a Raspberry Pi single-board computer with a radio transceiver unit. The Ethernet interface of the Raspberry Pi is used to send the magnetic field measurements to AuroraWatch UK. When the Raspberry Pi is accessed over the network with Secure Shell (SSH) a display and keyboard are not needed. The Raspberry Pi runs the Raspbian linux distribution. The receiving software is written in Python.

Figure 1.6: Base unit.

Part II

Construction

Chapter 2

Beginning construction

2.1 Anti-static precautions

2.2 Tools required

- Soldering iron.
- Side cutters.
- Small pliers.
- In-circuit serial programmer (ISP) for Atmel AVR microcontrollers. Instructions are given for the Atmel AVR Dragon but other programmers can be used.
- USB to TTL serial converter for 3.3 V operation, e.g., FTDI TTL-232R-3V3.
- Digital multimeter.
- Solderless breadboard (optional).

2.3 Order of assembly

For ease of access components should normally be fitted in order of increasing size, particularly increasing height. If this order is not observed it can be very difficult to access the pads of surface mount devices. It is also preferable that *passive* components (resistors, capacitors, inductors and crystals) are fitted before semiconductors (field-effect transistors, integrated circuits). This is because the semiconductors are easily damaged by electro-static discharge (sometimes this damage isn't immediately obvious). It is therefore more convenient to fit as many components as possible before fitting the semiconductors, at which point ESD precautions should be followed. As field-effect transistors are particularly vulnerable to damage by ESD it is recommended they are fitted as late as possible. From these guidelines the following order is recommended.

- Surface-mount passive components.
- Surface-mount semiconductors.
- Through-hole passive components.
- Through-hole semiconductors (FETs last).
- Switches.
- Connectors, battery holders.

The first PCB to be assembled is the FLC100 shield, this board provides the power to the system and will enable you to test each part correctly.

Chapter 3

FLC100 shield assembly

3.1 Introduction

The FLC100 shield is an Arduino "shield" which interfaces the microcontroller to the FLC100 fluxgate magnetometer sensor, if necessary translating logic levels. It is also responsible for generating the correct operating voltage for the microcontroller. There is more than one version of this board, be sure to use the instructions appropriate to your version.

3.2 FLC100 shield version 1.0

3.2.1 Description

The FLC100 shield version 1.0 operates at 3.3 V. **Do not attempt to use it with standard Arduino boards which are operated at** 5 V**.** The shield houses the XRF radio module, the boost power supply (which creates the 3.3 V supply for the microcontroller and radio) and the 3.3 V – 5 V level shifters.

There are both through-hole and surface mount versions of the boost power supply. It is suspected that the through-hole version causes RFI since the radio module often fails to receive messages. This problem was not apparent on the prototype board. The surface-mount version has no such problems and is the option which should be used. It is also cheaper and more efficient.

There is an option to fit a FLC100 sensor directly to the circuit board. This option is not used because the FLC100 is slightly temperature sensitive and better performance is obtained by positioning the sensor below ground. Whilst the board provides an option to fit an MCP3424 ADC and MAX619 charge-pump power supply they are also fitted remotely, below ground, for reasons of temperature stability.

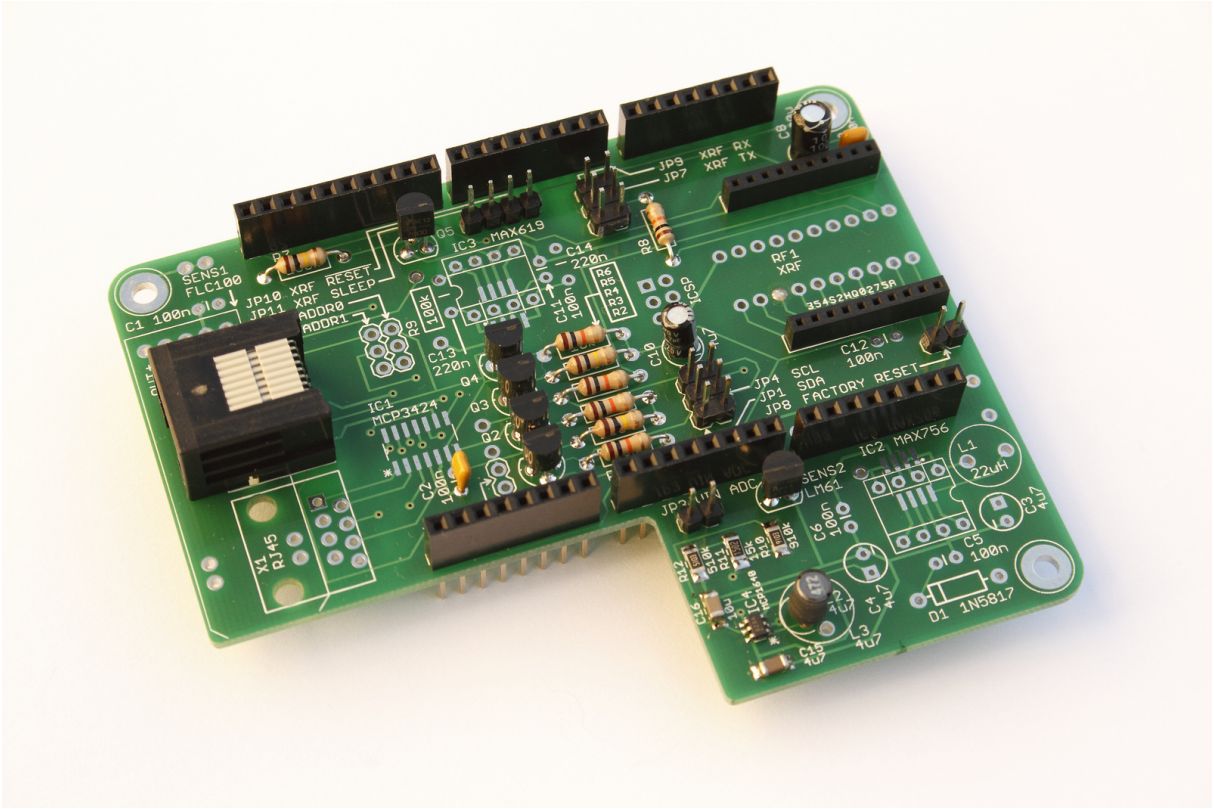


Figure 3.1: Completed FLC100 shield, except for fitting shunts onto the jumpers.

3.2.2 Order of assembly

1. IC4 (MCP1640).
2. R12 (510 kΩ).
3. R11 (15 kΩ).
4. R10 (910 kΩ).
5. C15 (4.7 µF).
6. C16 (10 µF).
7. 2 mm 10 way connectors for RF1. Ensure they are fitted flush to the PCB.
8. R1, R3, R4, R6, R8 (10 kΩ).
9. R2, R5, R7 (100 kΩ).
10. C2, C7, C9 (100 nF).
11. C10 (4.7 µF).
12. C8 (100 µF).
13. L2 (4.7 µH). The shorter lead should be connected to pin 1, which is the hole nearest the edge of the PCB. Although the orientation of inductors is normally ignored communication with the manufacturer revealed that the shorter lead indicates the start of the winding. This arrangement is preferred to help minimise RFI.
14. Stacking connectors, five 8 way and one 10 way. Ensure they are fitted flush to the PCB; solder one end first, then the other end. Only when you are happy they are flush should you solder the remaining pins.
15. JP1, JP4 (2×3 jumper).
16. JP7, JP9 (2×3 jumper).

17. JP10, JP11 (1×4 jumper).
18. JP3 (1×2 jumper).
19. JP8 (1×2 jumper).
20. X2 (RJ45 connector).
21. Modify the PCB by adding a link wire from the XRF ONSLEEP status pin to D23. See figure 3.2.
22. Q1, Q2, Q3, Q4, Q5 (2N7000).

Fit shunts to JP10, JP11, JP3. Fit shunts to JP7 and JP9, to the two connectors furthest from the edge of the PCB. **Do not fit the shunts so that they bridge between JP7 and JP9**.

If the microcontroller board has dedicated I2C connections (e.g.Calunium v2.0 or later), then no shunts should be fitted to JP1 and JP4. If the microcontroller has I2C connections in the standard Arduino Mega positions (e.g., Calunium v1.0, Arduino Mega, Arduino Mega 2560) then fit shunts to JP1 and JP4 to the two connectors closest to C10, otherwise fit shunts to JP1 and JP4 to the two connectors closest to the stacking connectors. **In no circumstances should the shunts bridge between JP1 and JP4**.

Leave JP8 open circuit. It is fitted only when the XRF1 radio module must be forced to use the default factory configuration.

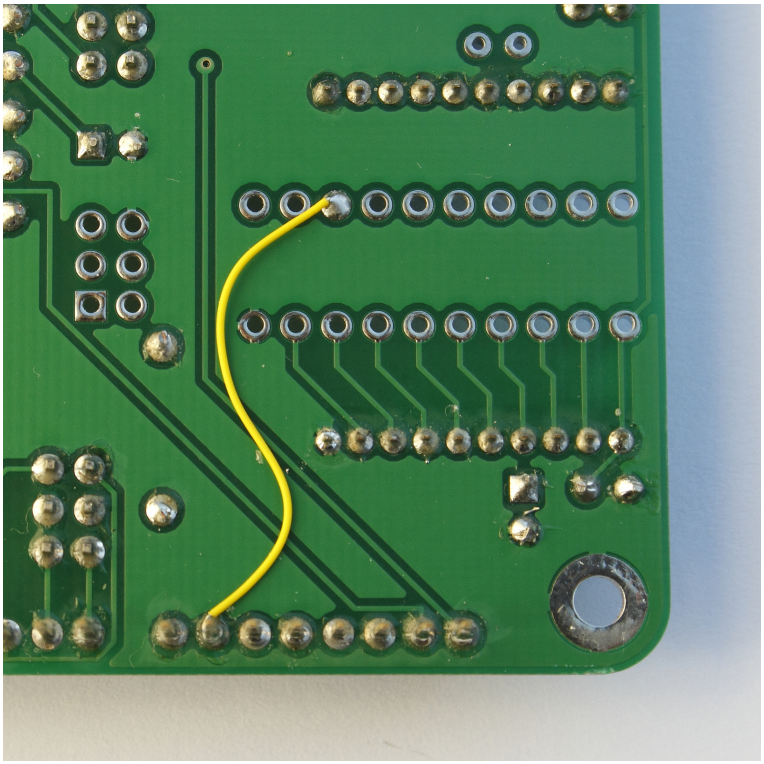


Figure 3.2: Modification to monitor sleep status of the XRF radio module.

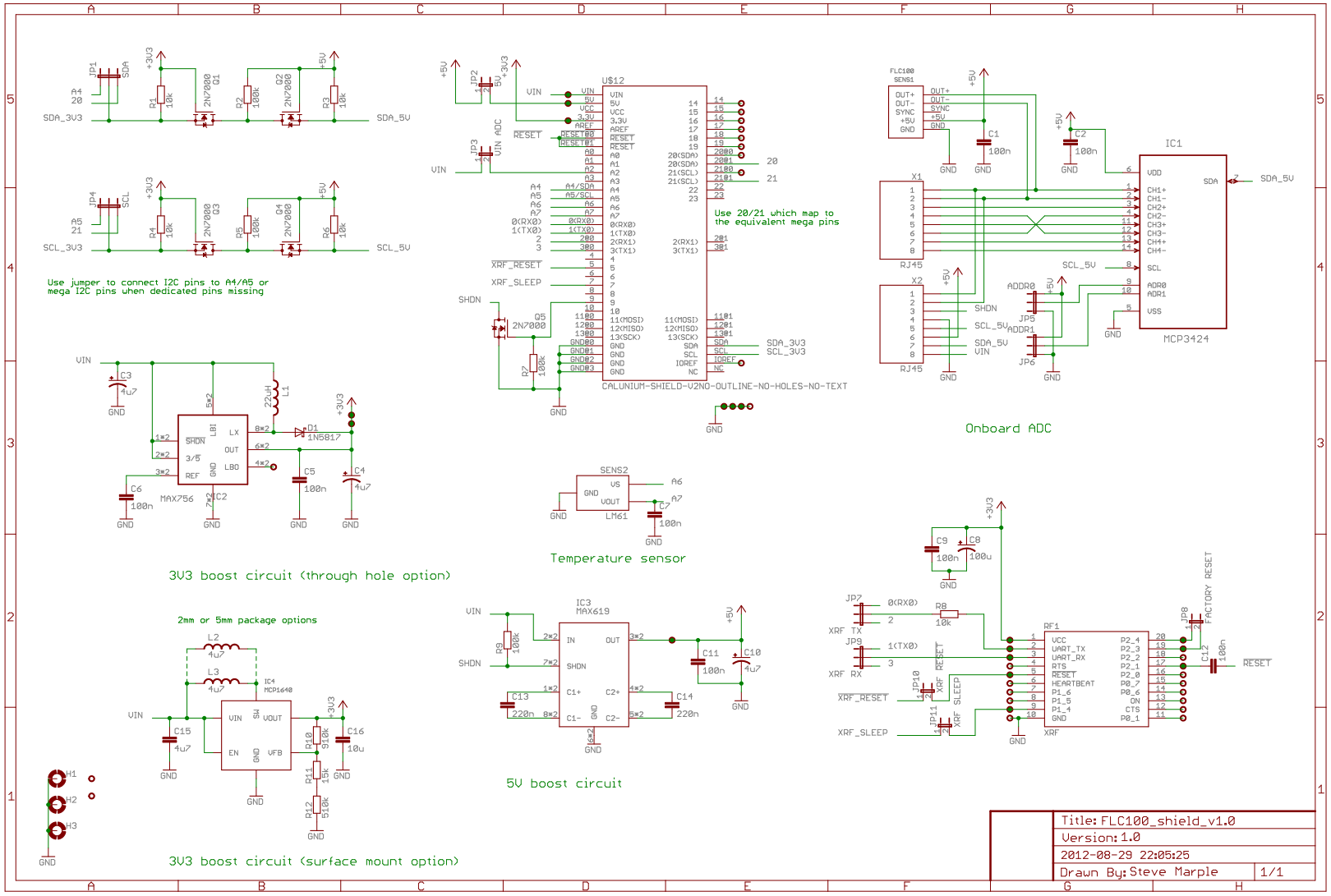


Figure 3.3: FLC100 shield v. 1.0 circuit diagram.

3.3 FLC100 shield version 2.0

3.3.1 Description

The FLC100 shield version 2.0 can operate at either 3.3 V or 5 V. This enables the board to be used either for battery-powered systems, with the microcontroller and radio operating at 3.3 V or with the standard Arduino Ethernet shield which requires the shield and microcontroller to operate at 5 V. When used in conjunction with the Arduino Ethernet shield it is assumed that the PoE module is fitted. The operating voltage chosen during construction determines which components are fitted.

The shield can also be used for cloud detection, for which it supports the operation of an MLX90614 non-contact IR thermometer, HIH61xx humidity and ambient temperature sensor, and Embedded Adeventures lightning sensor module. These functions are outside of the scope of this document and will not be described here.

3.3.1.1 Battery operation

When built for battery operation the shield houses the XRF radio module, the boost power supply (which creates the 3.3 V supply for the microcontroller and radio) and the 3.3 V – 5 V level shifters. The boost regulator can be built onto the PCB using discrete components. A simpler alternative which avoids the need to solder small surface mount components is to fit a ready-built third-party module.

3.3.1.2 Power over ethernet operation

If used in conjunction with the Ethernet shield the XRF radio module and boost power supply are not fitted. The Ethernet shield provides a 9 V output onto the `Vin` pin, requiring that a linear or buck regulator is fitted for 5 V operation.

3.3.2 Order of assembly

3.3.2.1 Battery-powered version

> Build instructions for the battery-powered option of the FLC100 shield version 2.0 are untested. Proceeed with caution.

Order of assembly for the battery-powered version.

If using the on-board surface-mount boost regulator fit:

1. IC4 (MCP1640).

2. R19 (510 kΩ).
3. R18 (15 kΩ).
4. R17 (910 kΩ).
5. C4 (4.7 µF).
6. C5 (10 µF).

For all battery-powered versions fit:

7. 2 mm 10 way connectors for RF1. Ensure they are fitted flush to the PCB.
8. R4 (1 kΩ).
9. R1, R3, R5, R7, (10 kΩ).
10. R2, R6 (100 kΩ).
11. R8 (1 MΩ).
12. C1, C3 (100 nF).
13. C2, C9 (100 µF).
14. C6 (100 µF 25 V).
15. L1 (4.7 µH). The shorter lead should be connected to pin 1, which is the hole nearest the edge of the PCB. Although the orientation of inductors is normally ignored communication with the manufacturer revealed that the shorter lead indicates the start of the winding. This arrangement is preferred to help minimise RFI.
16. Stacking connectors, five 8 way and one 10 way. Ensure they are fitted flush to the PCB; solder one end first, then the other end. Only when you are happy they are flush should you solder the remaining pins.
17. JP2 (1×2 jumper).
18. JP3, JP5 (1×3 jumper).
19. JP4, ISP header (2×3 jumper).
20. X1 (RJ45 connector).
21. Q1, Q2, Q3, Q4, Q5 (2N7000). Fit last to minimise risk of damage from ESD.

If using an external boost regulator fit:

22. Boost regulator module (Ciseco PowerPOD NCP1402 3V3).

3.3.2.2 Order of assembly: PoE version

STOP Build instructions for PoE option of the FLC100 shield version 2.0 are untested. Proceeed with caution.

Order of assembly for the PoE version.

1. R1, R3, R5, R7, (10 kΩ).
2. R2, R6 (100 kΩ).
3. C1, C3 (100 nF).
4. C7 (4.7 µF).

5. C2, C9 (100 µF).
6. C6 (100 µF 25 V).
7. L1 (4.7 µH). The shorter lead should be connected to pin 1, which is the hole nearest the edge of the PCB. Although the orientation of inductors is normally ignored communication with the manufacturer revealed that the shorter lead indicates the start of the winding. This arrangement is preferred to help minimise RFI.
8. Stacking connectors, five 8 way and one 10 way. Ensure they are fitted flush to the PCB; solder one end first, then the other end. Only when you are happy they are flush should you solder the remaining pins.
9. JP3 (1×3 jumper).
10. JP4, ISP header (2×3 jumper).

If the master PCB of the FLC100 remote sensor is version 2.0 and it will be fitted with a linear regulator (MCP1702) then JP5 can be omitted. Otherwise fit:

11. JP5 (1×3 jumper).
12. Add shunt to JP5, connect the centre pin to `Vin` if a linear regulator is fitted to the master FLC100 remote sensor PCB; connect to `+3V3` if the MAX619 boost regulator is used on the master FLC100 remote sensor PCB.

The connector to the remote sensor PCB(s) can be either RJ11 or RJ45. The RJ45 has the advantage that a standard (straight-through) ethernet cable can be used. However it has the disadvantage that it is possible to inadvertently fit the PoE ethernet connection into the wrong socket.

13. X1 (RJ45 connector) or X2 (RJ11 connector).

mplete Add shunts to jumper blocks.

14. Fit 3 shunts to JP4, in the positions marked on the silkscreen.
15. Fit a shunt to JP5, linking the centre pin with `+5V`.

> If JP2 is fitted **do not** fit a shunt when used for PoE operation, it will damage the microcontroller. Measurement of the input voltage can only be performed when $V_{in} \leq V_{cc}$.

Finally fit last to minimise risk of damage from ESD.

16. Q1, Q2, Q3, Q4 (2N7000).

3.3.2.3 Other variations

If is possible (though not desirable) for the 5 V supply for the remote sensor PCB(s) to be generated by the FLC100 shield. To enable this fit JP1 and add a shunt to link the 5 V rails between the PCBs.

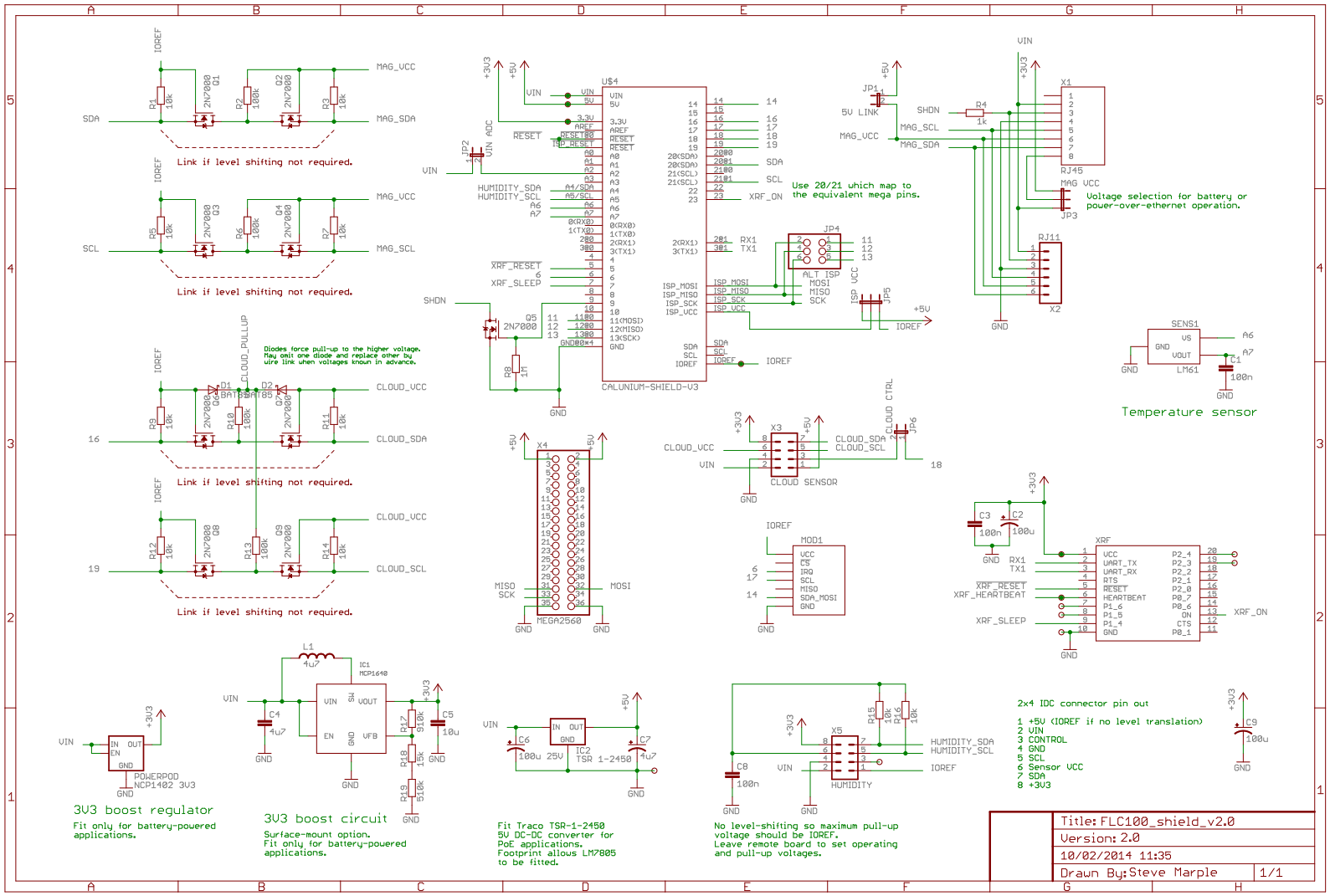


Figure 3.4: FLC100 shield v. 2.0 circuit diagram.

Chapter 4

Calunium assembly

4.1 Introduction

The Calunium microcontroller development board is intended to be a flexible system for both development and embedded use. As such it has various hardware options and careful attention must be paid to assembling it for optimum performance. Parts which are not needed are omitted to lower power consumption (e.g., power LED, USB controller).

4.2 Calunium version 2.0 and version 2.1

4.2.1 Order of assembly

Fit components in order:

1. IC2. The standard real-time clock is the MicroChip MCP90410 but MicroChip MCP79411 or MCP79412 can be used without any other changes. It is also possible to fit the Maxim DS1338-33 real-time clock, but see below for changes.
2. Y1 (32.768 kHz).
3. R1, fit a 680 Ω resistor. Ignore the 1 kΩ marking; a lower value resistor is used to enable the green LED to be seen more clearly in daylight.
4. R7 (1 kΩ). Do not fit if using the DS1338-33 real-time clock. Instead link between R7 and D2 at the end nearest the RTC battery, as indicated by the white line on the silkscreen. The wire will bypass both R7 and D2 which are not required for the DS1338-33.
5. R4, R5 (4.7 kΩ).
6. R3, R6 (10 kΩ).
7. L1 (10 µH).
8. 40 pin socket for IC4.
9. C4 (100 pF).
10. C3, C5, C7, C9, C12 (100 nF).
11. C2, C8 (1 µF).

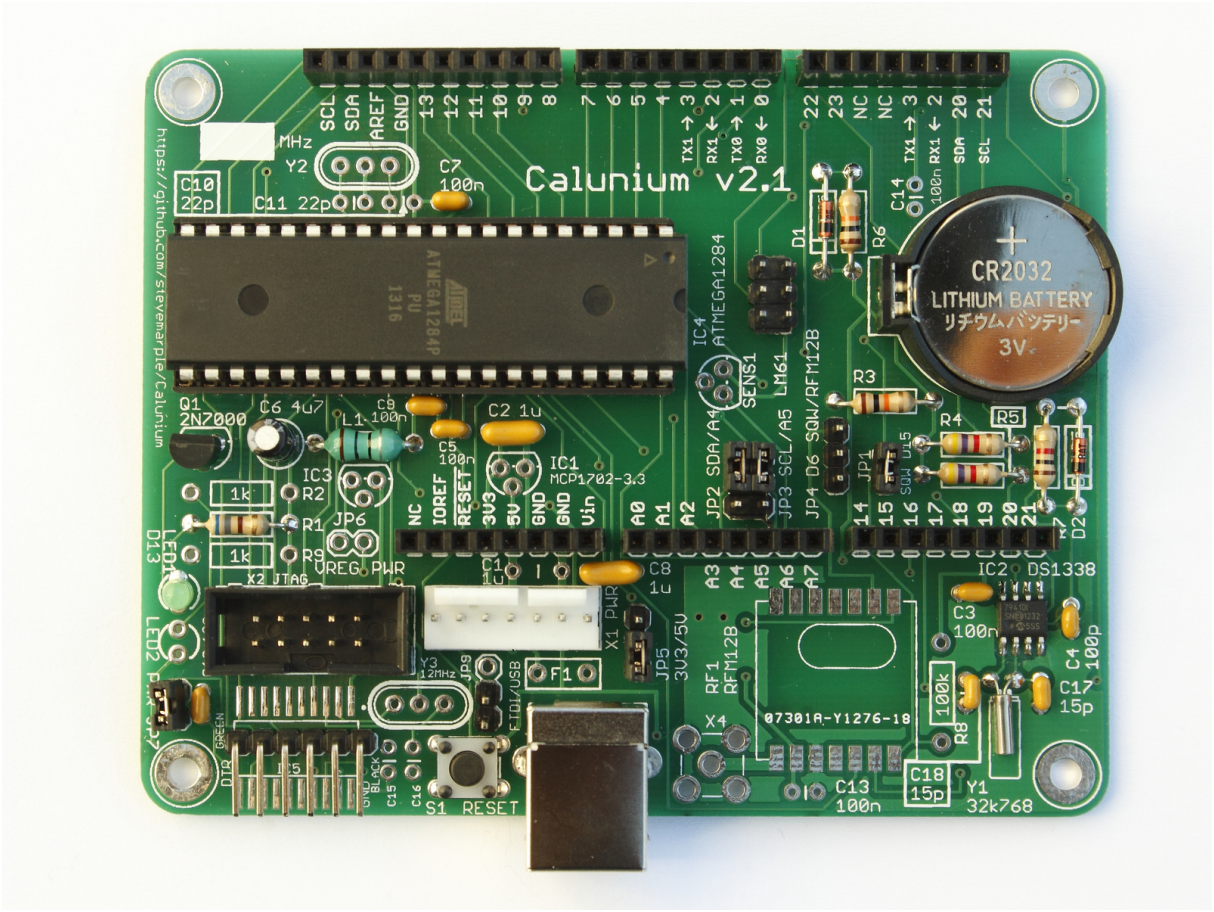


Figure 4.1: Completed Calunium v2.1.

12. D1, D2 (BAT85). Do not fit D2 if using the DS1338-33 real-time clock.
13. LED1 (green LED). The cathode is nearest LED2, see figure 4.2.
14. C6 (4.7 µF).
15. ICSP header (2 × 3 jumper block).
16. JP2 and JP3. Fit as combined 2 × 3 jumper block.
17. JP1, JP7 (1 × 2 jumper).
18. JP4, JP5 (1 × 5 jumper).
19. X5 (1 × 6 right-angle or vertical header for UART0).
20. S1 (reset switch).
21. Arduino headers. TO DO: Add description
22. C17, C18 (15 pF). Do not fit if using DS1338-33 RTC. For Calunium version 2.0 the capacitors must be fitted on the reverse side of the board (see Figure 4.3 as no specific mounting holes exist (error caused by using an earlier, incorrect datasheet which did not show the load capacitors).
23. X1 (Molex power header). Ensure correct orientation, with the backplate closest to the Arduino headers.
24. TO DO: Add component name (RTC battery holder).
25. Q1 (2N7000). This item is very sensitive to damage by electrostatic discharge!
26. Battery (CR2032). Check that the battery backup pin (3) on the RTC measures 3.0 V.
27. TO DO: Fit shunts to jumpers . . .

Do not fit the ATmega1284P microcontroller until after testing the board power supplies.

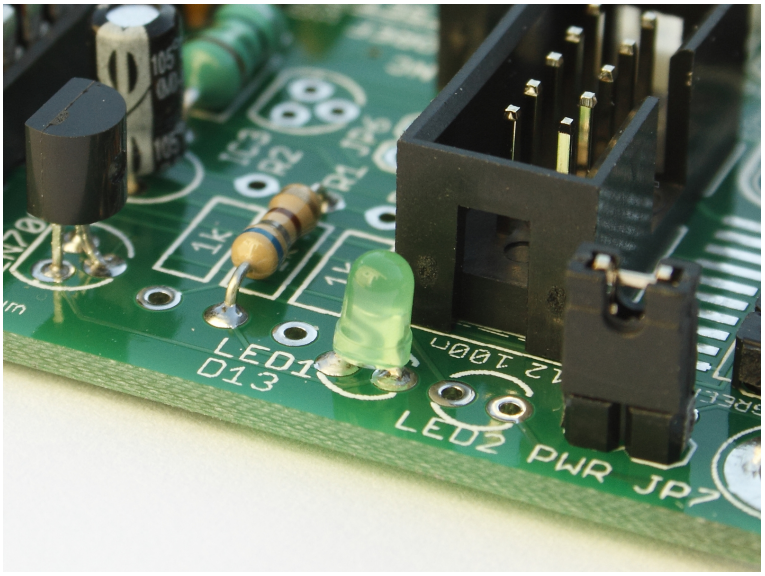


Figure 4.2: LED orientation.

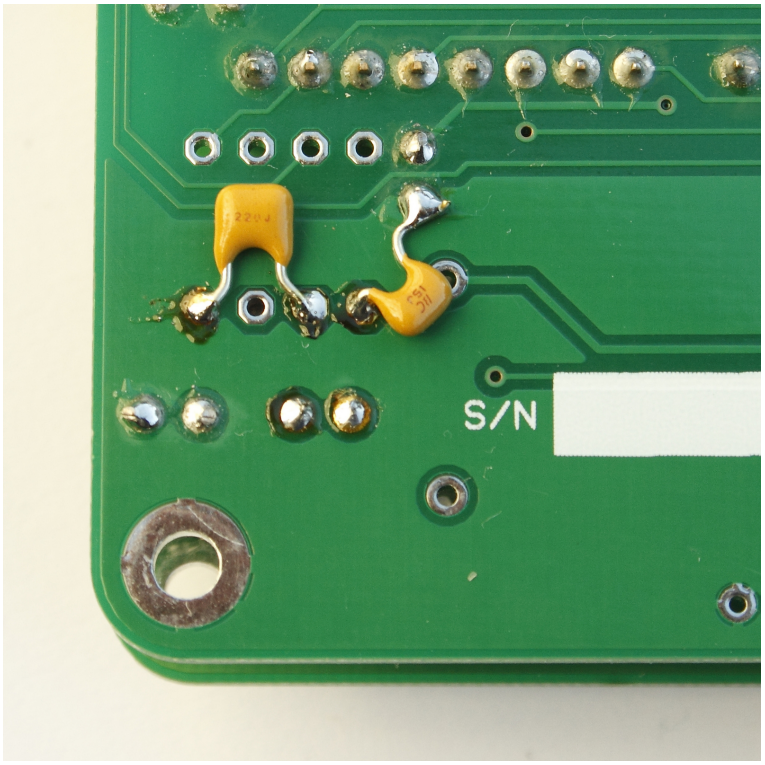


Figure 4.3: Real-time clock load capacitors (Calunium v 2.0).

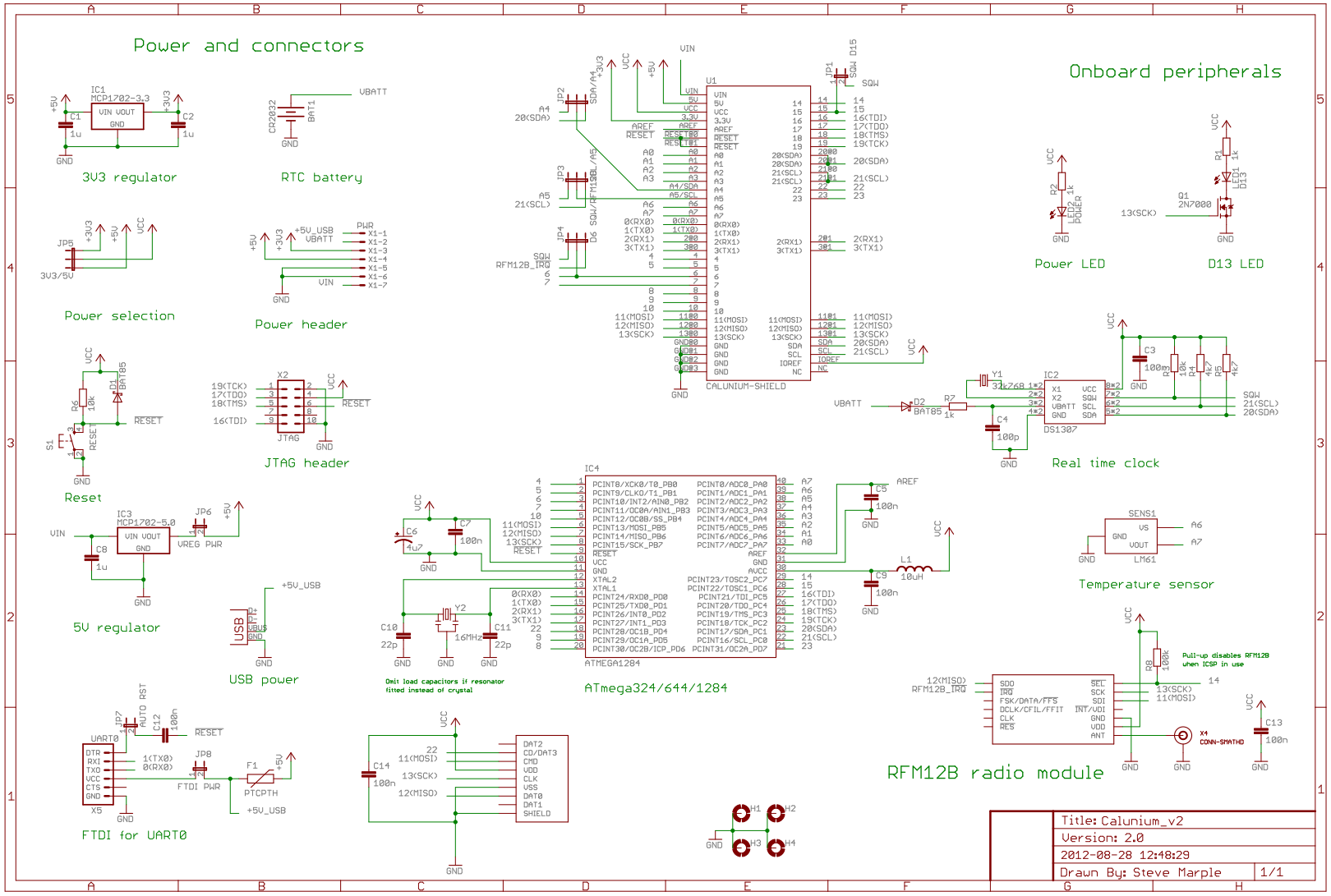


Figure 4.4: Calunium v. 2.0 circuit diagram.

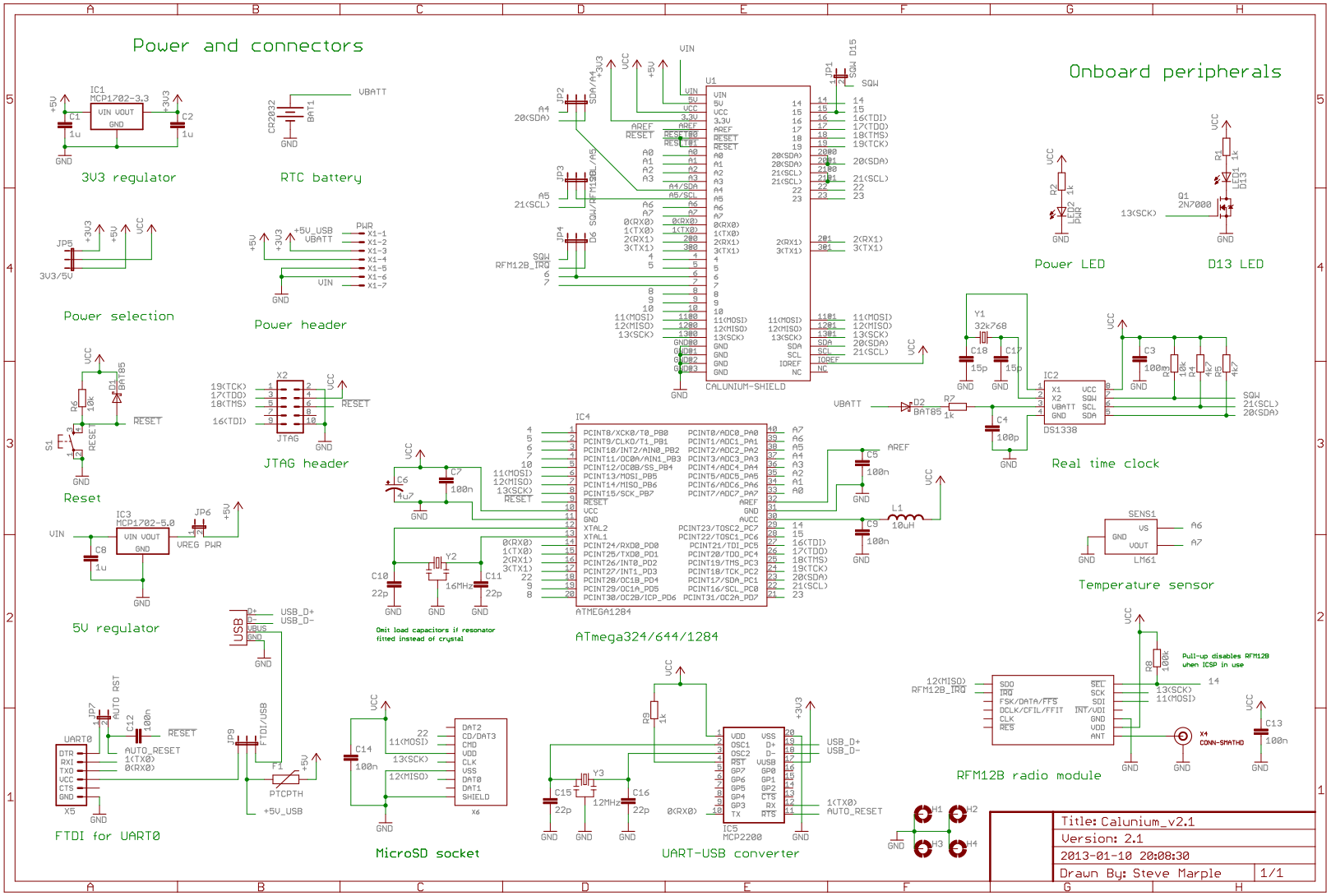


Figure 4.5: Calunium v. 2.1 circuit diagram.

4.3 Testing the board

4.4 Programming the firmware

These instructions assume you are using the Atmel AVR Dragon in ISP mode but adapting them to suit your programmer should be straightforward; see the `avrdude` manual page for further information.

4.4.1 Programming the bootloader

Power up the Calunium board and connect the programmer. Ensure the cable is correctly orientated at both ends. The bootloader can be compiled and programmed simply, as user `pi`: TO DO: Check directory

```
cd /home/pi/xboot
make clean
make SHELL=bash calunium_8MHz_RC_ISP.conf.mk program
```

TO DO: ignore lock bit verification errors.

If the bootloader is correctly programmed the green LED connected to D13 on the Calunium PCB should flash at about 1 Hz.

4.4.2 Programming the magnetometer firmware

Ensure that the shunt marked “AUTO RST” is fitted, and that the shunt marked “FTDI PWR” is omitted. Connect the FTDI cable and identify the USB device file; as user `pi`:

```
dmesg | tail
```

Look for a line containing text similar to

```
FTDI USB Serial Device converter now attached to ttyUSB0
```

For the case above the device file `/dev/ttyUSB0`. Now program the microcontroller using the xboot bootloader. Replace `/dev/ttyUSB0` with the device file one your system. As user `pi`:

```
cd /home/pi/AuroraWatchNet/firmware/magnetometer
avrdude -p atmega1284p -b 38400 -c avr109 -P /dev/ttyUSB0 \
    -U flash:w:xrf_rf12-0.10a.bin:r
```

ℹ Whilst it is possible to program the firmware using the AVR Dragon alone this approach ensures that the xboot bootloader is present and functions correctly, allowing the microcontroller firmware to be updated over the radio link.

4.4.3 Generating the EEPROM settings

Correct operation requires that the settings and the HMAC-MD5 key used for signing messages are uploaded to the EEPROM in the ATmega1284P microcontroller. To generate the EEPROM image use the `generate_eeprom_image` program, as user `pi`:

```
/home/pi/AuroraWatchNet/software/server/awnetd/generate_eeprom_image.py  \
--file /home/pi/eeprom --site-id XX --sampling-interval-16th-s 480  \
--max-message-no-ack 60 --max-messages-led 30 --num-samples 15  \
--aggregate 1 --all-samples 0 --radio-type 0 --radio-xrf-channel 25  \
--mcp7941x-cal 0 --use-sd 0
```

Replace `XX` with the site ID for your site. AuroraWatch UK maintains a list of unique site IDs; zero can be used for testing.

These settings will cause the magnetometer to make measurements every 30 s.

ℹ Some settings are defined in terms of samples, rather than time. If the sample interval is altered the effective interval for these settings will also change. If the number of messages without an acknowledgement exceeds that defined by `--max-message-no-ack` (60 in the above example) then the microcontroller is rebooted in the hope of recovering communication. Similarly, the number of messages before the LED is switched off (see section 8.2) is defined by `--max-messages-led` (30 in the example above).

These settings are defined by number of messages to avoid problems should the system time be automatically corrected.

ℹ The `generate_eeprom_image.py` program generates a new random HMAC-MD5 key each time it is run. Use the `--hmac-key` option to preserve a previous HMAC-MD5 key setting.

4.4.4 Uploading the EEPROM settings

The EEPROM settings defined in the previous section can be uploaded directly, using ISP or JTAG programmers, or indirectly via the `xboot` bootloader. The upload can be performed as user `pi`, but if problems accessing the programmer device file is experienced it can be uploaded as user `root`.

To upload using the AVR Dragon in ISP mode:

```
avrdude -P usb -p atmega1284p -c dragon_isp \
-U eeprom:w:/home/pi/eeprom.bin:r
```

To upload using the AVR Dragon in JTAG mode:

```
avrdude -P usb -p atmega1284p -c dragon_jtag \
-U eeprom:w:/home/pi/eeprom.bin:r
```

To upload via the `xboot` bootloader:

```
avrdude -p atmega1284p -P /dev/ttyUSB0 -c avr109 -b 38400 \
-U eeprom:w:/home/pi/eeprom.bin:r
```

If the FTDI adaptor is connected to a different device file substitute `/dev/ttyUSB0` with the correct device file.

Chapter 5

Sensor PCB assembly

5.1 Sensor PCB version 1.2

5.1.1 Order of assembly

Fit components in order:

1. Turned pin sockets for the FLC100 sensor. Accurate alignment is important so use a peice of solderless breadboard to hold the male turned-pin headers (Figure 5.3). Insert the headers so that the conical part is pointing downwards. Fit the upside down turned-pin sockets onto the headers (Figure 5.4). Place the PCB onto the upside-down sockets (Figure 5.5) and solder all 7 connections. Remove from the breadboard, leaving the male headers in place. Carefully position the FLC100 sensor onto the male turned-pin headers. The top-side of the FLC100 has two yellow capacitors and the letters BS the circuit board. Solder the FLC100 sensor to the header. Gently remove the FLC100 sensor and place in an anti-static bag.
2. IC1 (MCP3424).
3. IC2 (MAX619) if using the SOIC option.
4. R1, R2, R3 (10 kΩ).
5. R4 (100 kΩ).
6. R5 (4.7 kΩ).
7. C1, C2, C5, C8 (100 nF).
8. C9 (10 nF).
9. C6, C7 (220 nF).
10. C3, C4 (4.7 µF).
11. D1 (BAT85).
12. IC socket for IC2 if using the DIP option.
13. SENS2 (LM61).
14. JP5 and JP7 (fit as 2×3 male header).
15. X1 (RJ45 vertical jack).
16. Fit IC2 (MAX619) into its IC socket if using the DIP option.
17. Q1 (2N7000). This item is very sensitive to damage by electrostatic discharge! This item

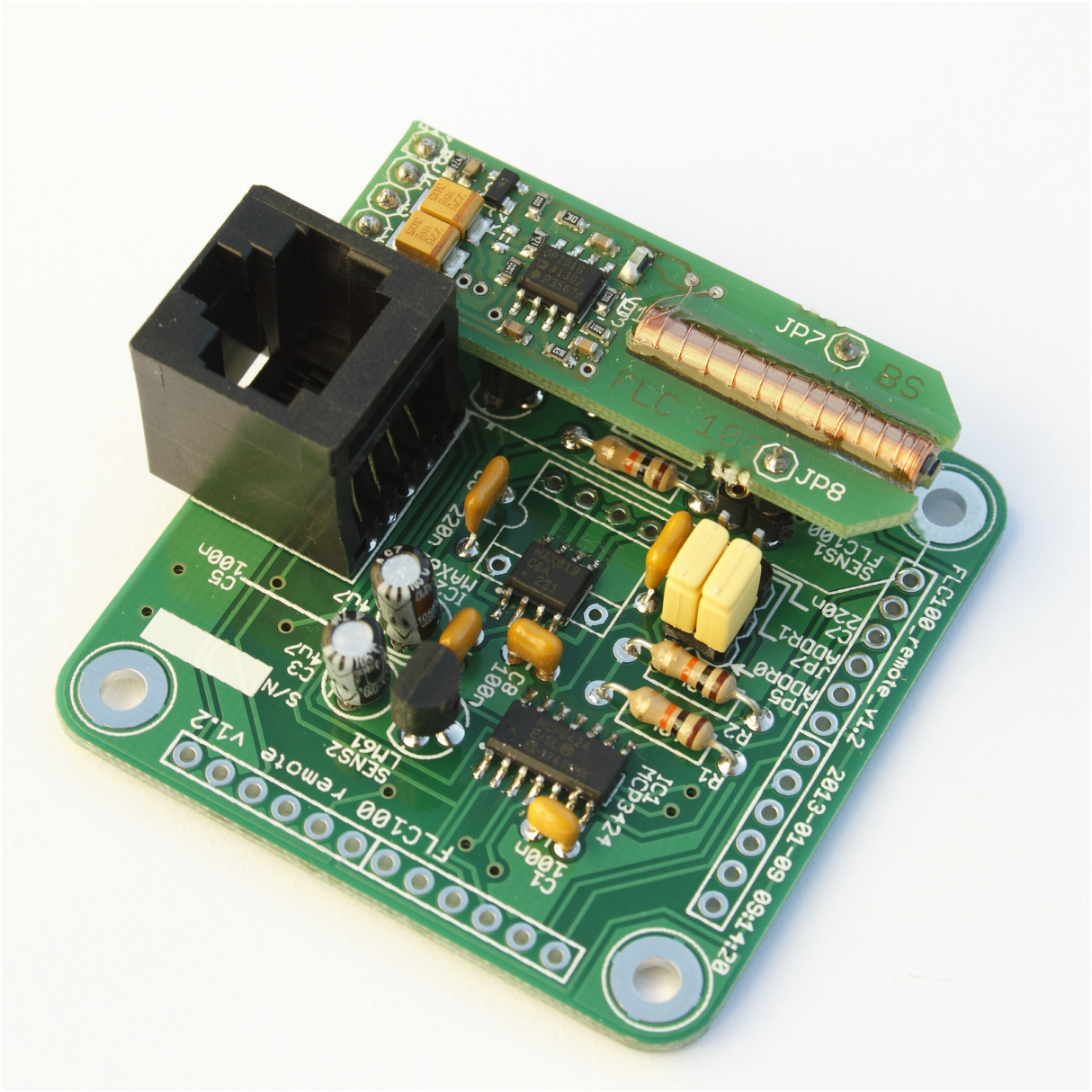


Figure 5.1: Completed sensor PCB (single-axis).

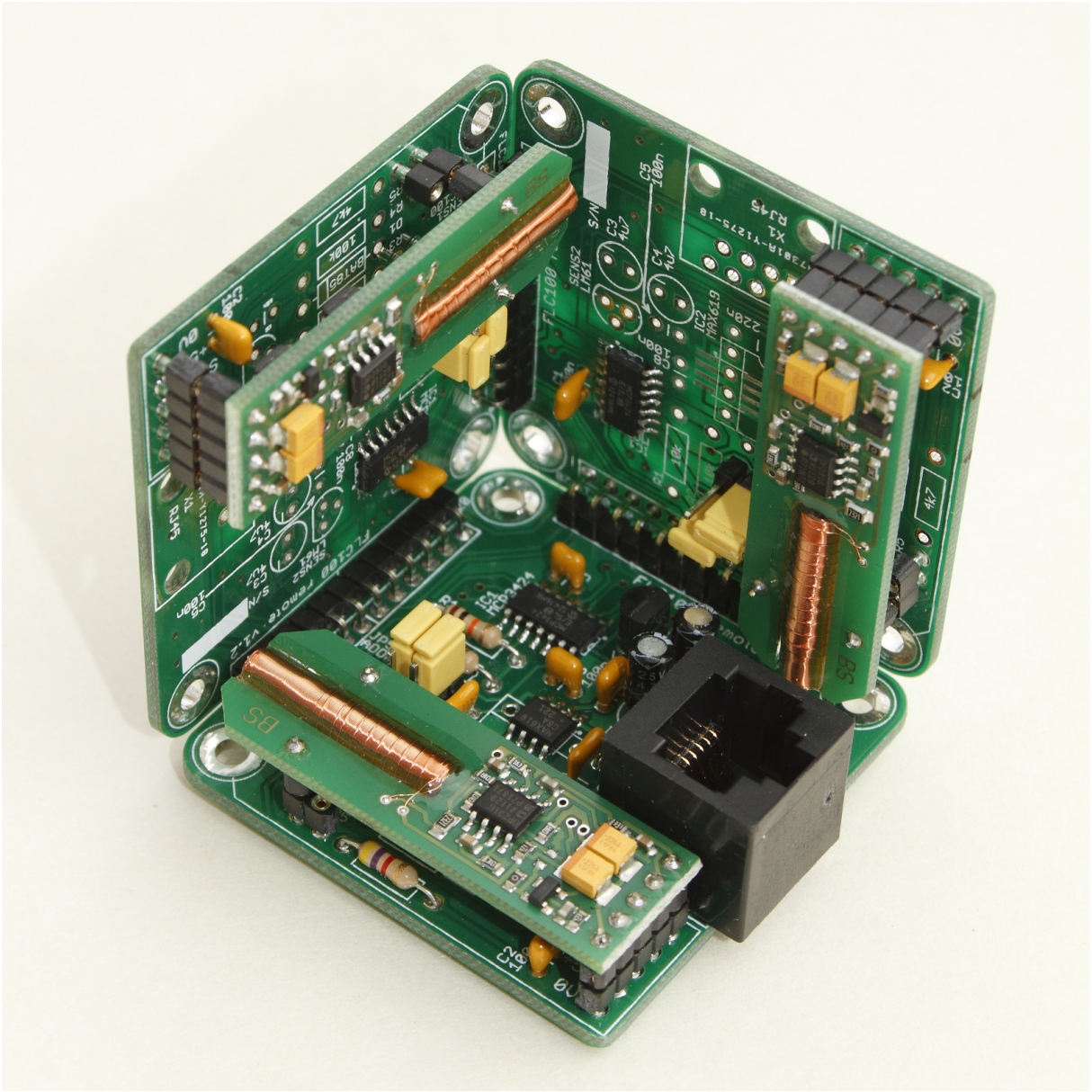


Figure 5.2: Completed sensor PCB (three-axis).

TO DO: Take photo and insert

Figure 5.3: Fit the male turned-pin headers.

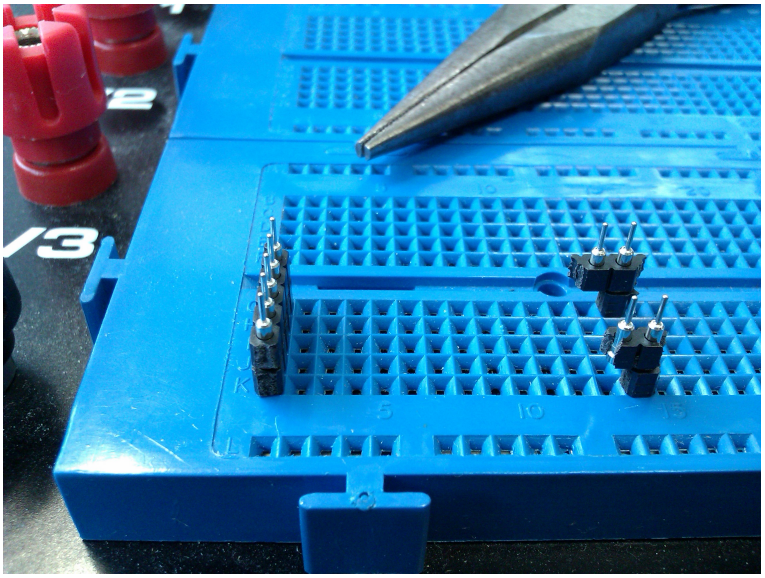


Figure 5.4: Fit the female turned-pin sockets.

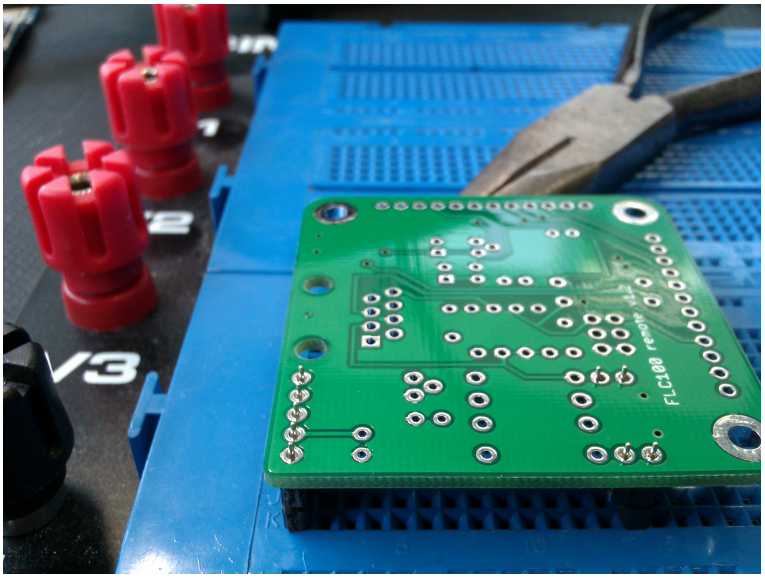


Figure 5.5: Place the sensor PCB onto the female turned-pin sockets. The pliers are used to support the other side of the PCB.

must be fitted close to the PCB to avoid fouling the FLC100 which is fitted over it.

18. SENS1. Gently fit the FLC100 sensor into the turned-pin sockets. Apply pressure only on the circuit boards, not the components or coil.

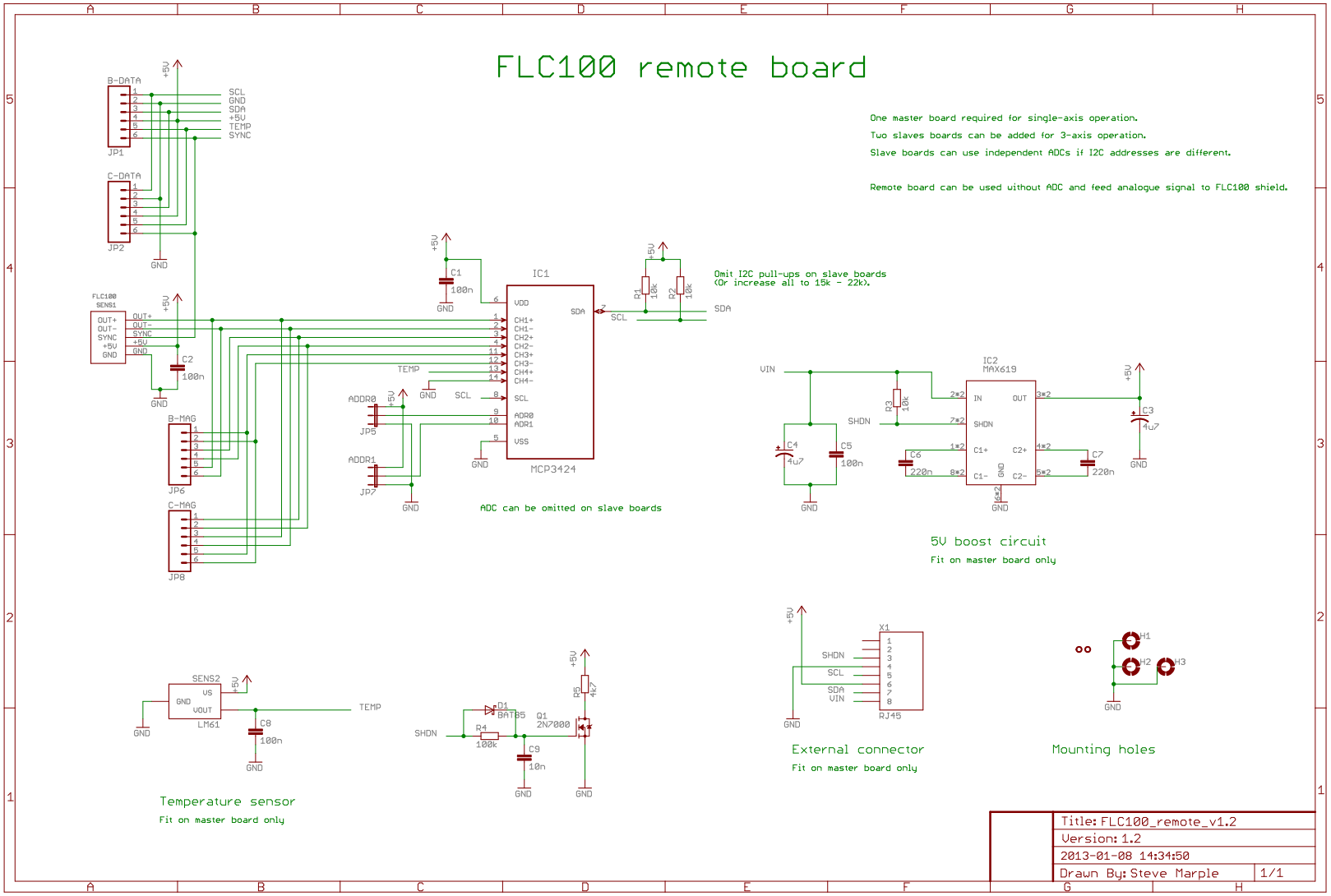


Figure 5.6: Sensor PCB version 1.2 circuit diagram.

Part III

Installation

Chapter 6

Site requirements

6.1 Sensor unit requirements

The sensor unit should be located outside away from human distrubances. The site for the sensor unit should be chosen with regard to the following requirements, with the highest priority given first.

- Within range of the base unit. For systems using radio communication this is likely to be less than 50 m. For systems using PoE the distance is limited by the maximum Ethernet segment length, 100 m.
- Away from moving metal objects, for example, trains (more than 50 m), cars (more than 20 m) and garage doors.
- Away from static metal objects, in particular those containing the *ferro-magnetic* materials iron, nickel and cobalt.

The FLC100 fluxgate magnetometer sensor is slightly sensitive to temperature variations. To ensure correct behaviour a stabilised temperature environment is required. This is obtained by burying the sensor.

6.2 Base unit requirements

The Raspberry Pi requires mains power and a wired network connection. Wireless networking is also possible but has not been tested; follow the instructions on the Raspberry Pi website. The base unit should be located indoors. Systems using radio communication will benefit from the base unit being located as close to the sensor unit as possible.

6.2.1 Network requirements

The following network requirements are needed for the system to operate fully:

- DNS resolution. This is normally provided as standard on most networks.
- Outgoing access on port 80 (HTTP) and 443 (HTTPS). Required for software updates and data transfer to AuroraWatch UK and the Met Office Weather Observations Website.
- Outgoing access on port 123 (NTP), or access to a local NTP server.
- Outgoing SSH access (port 22) is required if `rsync` or `rrsync` data transfers are enabled.

Chapter 7

Raspberry Pi setup

7.1 SD card creation

If your SD card already contains Raspbian you can skip to section 7.2.

Download the latest Raspbian image and copy to the SD card following the instructions on the Raspberry Pi web site. **Copying the compressed image to a FAT partition on the SD card will not work**.

These instructions assume Debian 8 ("Jessie") is used. Most users should probably download the desktop version. Advanced users, and those planning to use the Pi without a display connected ("headless") should probably download the minimal version without the desktop software.

7.2 Configuring Raspbian

> In the command window you can press the Tab key to have Linux complete the command or filename.

> If you are not familiar with the `nano` text editor read The Beginner's Guide to Nano, the Linux Command-Line Text Editor

Raspbian is most easily configured by booting the new image. If you are able to discover the IP address (for instance, by checking the DHCP tables of your home router) you can do this over the network using SSH. Otherwise you must use attach a keyboard and monitor to the Raspberry Pi. If you are familiar with Linux it is also possible to edit the files by mounting the SD card on another Linux system.

7.2.1 Raspbian configuration

Log in as `pi` and run

```
sudo raspi-config
```

7.2.1.1 Change user password

If the default password has not been changed then do so now to keep your system secure.

7.2.1.2 Hostname

If you wish change the hostname of your Raspberry Pi select `Advanced Options` and then `Hostname`.

If you have multiple Raspberry Pi computers on your network then you should arrange for them to have unique hostnames. Our preference is to set the hostname to `awn-xxx`, where `xxx` is the abbreviation (typically 3 letters) used for the magnetometer site.

7.2.1.3 Boot options

Configure the Pi for console access (text console, requiring the user to login). The console can be started manually when required with the `startx` command which saves memory when an interactive login is not required.

7.2.1.4 Localisation options

`cron` uses local time and the shift to and from daylight saving time complicates the `cron` tables. Set the Raspberry Pi's timezone to UTC to avoid daylight saving.

Select `Localisation Options` and then `Change Timezone`. For geographic area select `None of the above`, then select `UTC`.

7.2.1.5 Interfacing options

If you plan to log into the Raspberry Pi remotely you should enable the SSH server. If you will only log in via keyboard and monitor connected directly to the Pi then you can choose to disable the SSH server.

If the Raspberry Pi is to be fitted with the Ciseco *Slice of Radio* module then the serial interface must be enabled and the login shell over the serial port must be disabled.

7.2.1.6 Memory split

Select `Advanced Options` and then `Memory Split`. Set the GPU memory to `16` (MB).

7.2.1.7 SSH server

If you plan to log into the Raspberry Pi remotely you should enable the SSH server. If you will only log in via keyboard and monitor connected directly to the Pi then you can choose to disable the SSH server. Select `Advanced Options` and then `SSH`. Choose the appropriate option.

7.2.1.8 Expand Filesystem

The filesystem should be expanded to use all of the (micro)SD card. Advanced users planning to make a backup of the card may wish to perform this step last so that the backup can be smaller.

Select `Advanced Options` and then `Expand Filesystem`. Choose `Finish` and then reboot.

7.2.2 Configure proxy server

Not all networks require a proxy server (or web cache) to be used, your network adminstrator should be able to advise. If it is necessary the setting should be configured in two places.

As user `root`

```
nano /etc/environment
```

At the end of the file add a line similar to

```
http_proxy='http://proxyhost:port/'
https_proxy='http://proxyhost:port/'
```

You must replace `proxyhost` and `port` with the correct settings for your network. If the proxy server requires a username and password the lines should be similar to

```
http_proxy='http://username:password@proxyhost:port/'
https_proxy='http://username:password@proxyhost:port/'
```

Replace `username` and `password` with the values your network administrator has provided.

Repeat for the procedure, as `root`

```
nano /etc/apt/apt.conf.d/10proxy
```

Add a line similar to

```
Acquire::http::Proxy "http://proxyhost:port";
```

Or, if a password is required, similar to

```
Acquire::http::Proxy "http://username:password@proxyhost:port";
```

A separate line for HTTPS is not required in `/etc/apt/apt.conf.d/10proxy`.

Proxy settings will not take effect until you log out and log back in. Type

```
logout
```

and then log back in.

7.2.3 Upgrade installed software

As user `root`

```
apt-get update
apt-get upgrade
apt-get dist-upgrade
```

7.2.4 Remove swap file

To prolong the life of the SD card a swap file is not used. As user `root`

```
apt-get remove dphys-swapfile
```

7.2.5 Remove Wolfram Engine

Wolfram Engine uses over 680 MiB and is not needed. Remove to save valuable space on the SD card. As user `root`

```
apt-get purge wolfram-engine
apt-get autoremove
```

ℹ Wolfram engine is not installed by default in the light version of Raspbian.

7.2.6 Install missing software packages

As user `root`

```
apt-get install screen git git-doc git-man \
    python-pip ipython python-matplotlib \
    python-scipy python-serial python-daemon python-lockfile \
    avahi-daemon dnsutils i2c-tools python-smbus python3-smbus ntp
```

7.2.7 Configure file system mount options

As user `root`

```
nano /etc/fstab
```

Find the line where the root file system is mounted, it will look similar to

```
/dev/mmcblk0p2  /               ext4    defaults,noatime  0       1
```

Change the mount options (`defaults,noatime` in the example above) so that the mount options are now `noatime,nodiratime`. The line should look similar to the one below.

```
/dev/mmcblk0p2  /               ext4    noatime,nodiratime  0       1
```

At the end of the `/etc/fstab` add the following lines:

```
# tmpfs for AuroraWatchNet temporary files. Files will be deleted on
# a reboot, which is desirable for the NTP status files.
tmpfs  /home/pi/tmpfs  tmpfs  rw,size=100k,nr_inodes=1k,noexec,nodev,nosuid,uid=pi,gid=pi,mode=1700  0  0
```

As user `root`

```
mkdir /home/pi/tmpfs
mount /home/pi/tmpfs
```

7.2.8 Automatically create symlinks for FTDI all-in-one

As user `root`

```
nano /etc/udev/rules.d/90-usb_serial.rules
```

Insert the following lines into the file if they are not present:

```
# Have symlinks based on serial number for FTDI devices. Used for
# awnetd_monitor
SUBSYSTEMS=="usb", KERNEL=="ttyUSB[0-9]*", ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+="tty_ftdi_%s{serial}"
```

7.2.9 Regenerate SSH host keys

As user `root`

```
cd /etc/ssh
rm ssh_host_*_key{,.pub}
ssh-keygen -A
```

7.3 Installing the AuroraWatchNet server software

7.3.1 Install the Git repository

As user `pi`

```
git clone --recursive https://github.com/stevemarple/AuroraWatchNet.git
git clone --recursive https://github.com/stevemarple/auroraplot.git
git clone --recursive https://github.com/stevemarple/Calunium.git
git clone --recursive https://github.com/stevemarple/xboot.git
mkdir ~/bin
. ~/.bashrc
cd ~/bin
ln -s ../AuroraWatchNet/software/server/bin/awnetd.py
ln -s ../AuroraWatchNet/software/server/bin/send_cmd.py
ln -s ../AuroraWatchNet/software/server/bin/log_ip
ln -s ../AuroraWatchNet/software/server/bin/upload_data.py
cd ~
```

7.3.2 Create data directory

As user `root`

```
mkdir /data
chown pi.pi /data
```

7.3.3 Create configuration file

If the magnetometer is to use the Ciseco XRF radio modules to communicate between the sensor unit and base unit then as user `root`

```
cp ~pi/AuroraWatchNet/software/server/ini_files/calunium_xrf_awnet.ini /etc/awnet.ini
nano /etc/awnet.ini
```

TO DO: Add support for serial and USB versions.

If the magnetometer is to use Ethernet to communicate between the sensor unit and base unit then first identify which Ethernet shield is in use.

If using the original Arduino Ethernet shield, which is fitted with the WIZnet W5100 Ethernet controller, then as user `root`

```
cp ~pi/AuroraWatchNet/software/server/ini_files/calunium_w5100_awnet.ini /etc/awnet.ini
nano /etc/awnet.ini
```

If using the later Arduino Ethernet shield 2, which is fitted with the WIZnet W5500 Ethernet controller, then as user `root`

```
cp ~pi/AuroraWatchNet/software/server/ini_files/calunium_w5500_awnet.ini /etc/awnet.ini
nano /etc/awnet.ini
```

In the editor find the `[DEFAULT]` section, edit the `site` to the correct value. Navigate to the `[upload]` section and enter the correct values for `url`, `username`, `password` and `realm`. Ensure the leading comment character (`#`) is removed.

7.3.4 Configure `cron`

As user `pi`

```
crontab ~/AuroraWatchNet/software/server/crontabs/awnetd.crontab
```

If you plan to use the FTDI-all-in-one programmer and switch to indicate periods of bad data then uncomment the line with the reference to `awnetd_monitor.py`.

If the Raspberry Pi is using Wi-Fi networking uncomment the line referencing `network_watchdog`; it will cause the Raspberry Pi to be rebooted if it appears that Wi-Fi networking has stopped working for some reason.

The `log_ip` lines in the cron file cause the Raspberry Pi to periodically report that it is operating. This helps the AuroraWatch administrators monitor which stations are active. These reporting commands can be omitted if you prefer (comment out the command by inserting a `#` at the start of the line).

7.3.5 Configure `ifplugd`

Configure `ifplugd` to report when the network interface has been assigned an IP address, which helps the AuroraWatch administrators monitor which stations are active. This step can be omitted if you prefer. As user `root`

```
cd /etc/ifplugd/action.d
ln -s /home/pi/AuroraWatchNet/software/server/bin/log_ip
```

7.3.6 Configure python

Create the python local site directory and create appropriate symbolic links. As user `pi`

```
~/AuroraWatchNet/software/server/bin/setup.py --sudo
~/AuroraWatchNet/software/server/bin/setup.py --sudo
~/AuroraWatchNet/software/server/bin/setup.py --sudo
```

The first time the command runs there will be some errors printed as symbolic links are other configuration details are missing. On the second run confirm that the errors have been corrected.

7.3.7 Configure `ntp`

Check that the current time is correct by typing

```
date --utc
```

This will output the current date and time in UTC. If you aren't certain what the current time in UTC is then you can check at this web page, `https://www.google.co.uk/#q=utc+time`.

Check that the NTP service is running correctly:

```
~pi/bin/check_ntp_status --log-level=info
```

The last line should indicate "NTP synchronized". If it indicates that NTP is not synchronized consult your network manager for the correct NTP settings on your network.

7.4 Backup of SD card and expand filesystem

If you did not expand the (micro)SD card earlier then make a backup (if you wish) and run

```
sudo raspi-config
```

Select `Expand Filesystem`. Choose `Finish` and then reboot.

Chapter 8

Installation procedure

8.0.1 Tools required

- Spade.
- Fork.
- Small bucket or other container to remove soil from the bottom of the hole.
- Compass.

The following items are optional but if they are available they may be useful for digging the hole.

- Soil auger.
- Post hole digger.

8.1 Base unit installation

Connect the Raspberry Pi to wired ethernet connection. Connect the keyboard, mouse and monitor (if using) before powering up the Raspberry Pi. Connect the Raspberry Pi to a 5 V power supply with a suitable output current rating, this depends on which version of Raspberry Pi is to be used. If you are not using a monitor connect to the Raspberry Pi using SSH, the default hostname is `raspberry.local`.

8.2 Determining the maximum range for the radio link

If the sensor unit is to communicate over radio then before installing the sensor unit first check the maximum range. Note that this will vary according to many conditions and be sure to position the sensor well within the maximum range.

The maximum range can be determined with the following procedure:

1. Ensure that the Raspberry Pi is turned on and restart the recording daemon (p. 58). Monitor the recording process (p. 58).
2. Disconnect the remote sensor PCB from the FLC100 shield. Place the remaining circuitry about 5 m away from the Raspberry Pi. Keep the two units at least 2 m apart at all times to prevent overloading the radio receivers.
3. Connect the battery. The green LED on the Calunium PCB will flash 3 times with a frequency of about 1 Hz to indicate that the bootloader is waiting for data. The LED will then turn off.
4. About a second later the LED should light again, indicating the start of data transfer to the Raspberry Pi. If an acknowledgement is received the LED will turn off. This process should happen quickly, so that the LED flashes on momentarily. Just after the LED turns on the Raspberry Pi should output the received data message, along with its response to the sensor unit. If the LED remains on it indicates the microcontroller has not received an acknowledgement; see section 8.2.

   This step repeats itself following every data sampling operation, normally every 30 s. The time interval can be reduced by altering the sampling period (section B.2.9). A functioning radio communication link is required to alter the sampling period.
5. Move the sensor unit to its intended location. Try to position it at the same height as it will be when buried, approximately 10 cm above ground. Keep the radio antenna vertical. If the LED continues to flash briefly every 30 s the radio link is operating correctly. If not then find another location.
6. If the radio link appears to work correctly at the desired location then aim to find the maximum range by steadily increasing the distance between the sensor unit and base unit. Remember that the link is only tested after each sampling period. If the LED remains on move the sensor unit closer to the base unit and wait for the link to re-establish, indicated by the LED switching off.

> ℹ To save power the LED indicates the link status only for approximately 15 minutes after the power has been connected or the reset button pressed. If the LED has stopped indicating the link status press the reset button on the Calunium PCB.

8.2.1 Factors which alter the maximum range

The following factors can influence the range of the radio link:

- Obstructions. Try to find locations for the sensor unit and base unit which have direct line of sight. It is acceptable for walls to be in the path but note they will attenuate the signal. Avoid raised ground whenever possible. Placing the base unit on the first or second floor is likely to give improved range.

- Sources of radio noise. Some electrical equipment may cause RFI, which will reduce the range of the radio link. Try to keep both the sensor unit and base unit away from other electrical equipment, particularly those containing radio transmitters. “Smart” electric meters can contain radio transmitters which operate on the same 868 MHz ISM radio band.

8.3 Installing the sensor unit

The sensor unit must be should be buried to a depth of 0.85 m. If it is buried deeper then the radio antenna may be too close to the ground and compromise the wireless communication range, if it is buried too shallow the unit will be more susceptible to temperature variations.

After digging a suitable hole install the enclosure as vertical as possible and backfill the hole. Insert the wooden frame and rockwool into the enclosure. Using a compass align the north arrow on the wooden frame to point towards magnetic north. Connect the RJ45 cable to the FLC100 shield. Connect the power lead. Fit two D cell batteries into the battery holder and place on the wooden frame. Press the reset button and observe the green LED. It should flash 3 times to indicate the bootloader operation and then intermittently to indcate successful data transmission; see section 8.2.

Chapter 9

Contributing data

9.1 Contributing data to AuroraWatch UK

9.1.1 Use of data by AuroraWatch UK

Data contributed to AuroraWatch UK will be combined with other magnetometer data for the purpose of generating AuroraWatch UK or other auroral-related alerts. In future the alerts data are likely to be made available via a public API under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 license. Ideally you will also license the magnetometer data under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 license, remembering to define your own attribution requirements.

> If you choose a more permissive license, such as without the attribution, and/or non-commercial clauses, but retain the share-alike clause you must grant AuroraWatch UK permission to use the data to generate alerts under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 license. This is because the share-alike clause restricts others from imposing additional restrictions.

The magnetic field data collected by AuroraWatch UK magnetometers will be made publically available under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 license, with a short embargo period (24 to 48 hours). If your magnetometer data is also licensed as CC BY-NC-SA 4.0 then AuroraWatch UK will share your data in the same way.

9.1.2 Methods to upload data

Two methods to upload data to AuroraWatch UK are supported, using rsync through an SSH tunnel, or by HTTP. Rsync contains an algorithm to efficiently transfer only the differences btween the local and remote files and thus is ideal for transferring the real-time data files. SSH is used to provide a secure connection method. However, SSH access may not be possible on some

networks (e.g., school networks). For cases when rsync cannot be used a HTTP upload method is available which emulates some of the behaviour of rsync; whenever possible only the latest additions to a file are uploaded. The upload method is normally defined in `/etc/awnet.ini` configuration file, in the `[upload]` section.

9.1.2.1 Rsync uploads

As user `root` edit `/etc/awnet.ini`. At the end of the file add the `[upload]` section if it is missing, it should appear as

```
[upload]
method = rsync
```

As user `pi` create the keys for public key authentication:

```
ssh-keygen -t dsa
```

When prompted for the filename to save the key press  to accept the default. When prompted for the passphrase press for an empty passphrase. Keep the private key (`/home/pi/.ssh/id_dsa`) secret, send the public key (`/home/pi/.ssh/id_dsa.pub`) to AuroraWatch UK.

Create the SSH config file to define hostname and user used for data transfer. As user `pi` edit the file `/home/pi/.ssh/config`, it should look similar to

```
Host awn-data
Hostname uploadhost
User uploaduser
```

You will need to obtain the upload hostname and username from AuroraWatch UK.

Insert an instruction into the `crontab` file to upload the data at regular intervals. As user `pi`:

```
crontab -e
```

Add the following lines:

```
### rsync upload
*/3 * * * * nice /home/pi/bin/upload_data.py > /dev/null 2>&1
```

9.1.2.2 HTTP uploads

As user `root` edit `/etc/awnet.ini`. At the end of the file add the `[upload]` section if it is missing, it should appear as

```
[upload]
method = http
url = upload_URL
realm = upload_realm
password = upload_password
```

You will need to obtain the upload URL, realm and password from AuroraWatch UK.

Insert an instruction the the `crontab` file to upload the data as regular intervals. As user `pi`:

```
crontab -e
```

Add the following lines:

```
### HTTP upload
# Upload text data for today at regular intervals
*/5 * * * * nice /home/pi/bin/upload_data.py -s today --file-types awnettextdata
# Make several attempts to upload all files from yesterday
5 */6 * * * nice /home/pi/bin/upload_data.py -s yesterday > /dev/null 2>&1
```

9.2 Contributing data to the Met Office

Data can be contributed to the Met Office Weather Observations Website (WOW). First create a user account in WOW, see `https://register.metoffice.gov.uk/WaveRegistrationClient/public/register.do?service=weatherobservations`. Then register your site (`https://wow.metoffice.gov.uk/sites/create`) to obtain a site ID and 6 digit authorisation key.

The relevant information must be added to the `/etc/awnet.ini` file. As user `root`:

```
nano /etc/awnet.ini
```

Add the following lines:

```
# Upload to Met Office WOW
[wow]
url = <upload URL>
site_id = <site ID>
site_auth = <authorisation key>
```

Replace `<upload URL>` with the appropriate URL for magnetometer data. (This information has not yet been published). Replace `<site ID>` with the 36 character site indentifier and `<authorisation key>`) with the 6 digit authorisation key.

Data can be uploaded manually by running the following command as user `pi`:

```
/home/pi/AuroraWatchNet/software/server/bin/upload_wow.py -c /etc/awnet.ini -s yesterday
```

ℹ The WOW API does not provide a means to check if the data has been uploaded previously, and the WOW site does not check for or remove duplicate data. It is therefore necessary for the user to check if data has been already uploaded. For this reason it is recommended that data is uploaded only data recording to a given file has completed, i.e., avoid uploading data from the current day.

It is possible to automate the data uploads with a cron job. As user `pi`:

```
crontab -e
```

At the end of the crontab add the following lines:

```
# Upload data to Met Office WOW
5 * * * * /home/pi/AuroraWatchNet/software/server/bin/upload_wow.py -c /etc/awnet.ini -s yesterday > /dev/null 2>&1
```

Part IV

Operation

Chapter 10

Raspberry Pi operation

10.1 Introduction

For generic operation of the Raspberry Pi (setting the hostname, assigning a fixed IP address etc.) please see the Raspbian documentation, `http://www.raspbian.org/RaspbianDocumentation`.

10.2 Shutting down the Raspberry Pi

The Raspberry Pi must be shutdown cleanly before power is removed:

```
sudo shutdown -h now
```

Before removing the power wait until only the red power LED is lit; wait a further two seconds to ensure futher access to the SD card is not needed. If the power is removed whilst data is being written to the SD card it will corrupt the file system.

To reboot the Raspberry Pi use

```
sudo shutdown -r now
```

10.3 Starting and stopping the data recording daemon

Data is recorded on the Raspberry Pi using a *daemon* process, which is started and stopped by the Debian init scripts. The scripts must be started and stopped as user `root`, the actual data

recording process runs as user `pi`.

To start data recording

```
sudo /etc/init.d/awnetd start
```

To stop data recording

```
sudo /etc/init.d/awnetd stop
```

It is also possible to check the status of the data recording process

```
sudo /etc/init.d/awnetd status
```

The `restart` option forcibly stops recording (if running) and then starts it again:

```
sudo /etc/init.d/awnetd restart
```

10.4 Monitoring the data recording process

The data recording process directs its standard output and error streams to a virtual terminal using `screen`. It is possible to attach to this virtual terminal to monitor the output.

As user `pi`

```
screen -r awnet
```

To exit from `screen` type CTRL - a , d .

Pressing CTRL - c will terminate the recording process.

Chapter 11

Software and firmware updates

11.1 Raspbian updates

See Raspberry Pi web pages, `https://www.raspberrypi.org/documentation/raspbian/updating.md`.

11.2 AuroraWatchNet software updates

The software can be updated easily simply by *pulling* a new version from the Github repository. As user `pi`

```
cd ~/AuroraWatchNet
git pull
```

11.3 Sensor unit firmware updates

Only apply firmware updates if they are necessary. The firmware version selected must match the hardware, if it does not the sensor unit will be left inoperable and recovery will require an in-circuit serial programmer. First ensure that the AuroraWatchNet software has been updated (section 11.2) and the recording process restarted (section 10.3). To update the firmware

```
send_cmd.py --upgradeFirmware=name-version
```

Replace *name-version* with the firmware name/version string.

Appendix A

Configuration file options

A.1 Introduction

Many of the AuroraWatchNet programs read a common configuration file, typically located at `/etc/awnet.ini`. The configuration file is broken into sections, each of which starts with a section header in square brackets (`[like_this]`). Other lines contain key names and values, written as `key = value`. Leading and trailing whitespace around both the key and value is ignored. Section headers and key names do not contain whitespace, words may be separated with an underscore. The configuration file can also contain comments, which are entered with a hash (`#`) or semi-colon (`;`) as the first character.

The configuration file is parsed using Python's `SafeConfigParser` module. This allows values defined in the same section, or in the `[DEFAULT]` section to be inserted into other key value definitions. This feature is commonly used to insert the site abbreviation into filenames.

A.2 `[DEFAULT]`

The `[DEFAULT]` section is special as it defines values which can used elsewhere in the configuration file.

A.2.1 project

Define the project. This is used elsewhere within the configuration file, e.g., data filenames.

Default: none.

A.2.2 site

Define the site code, typically a three letter abbreviation. This is used elsewhere within the configuration file, e.g., data filenames.

Default: none.

Example:

```
site = lan1
```

A.3 [awnettextdata]

Options associated with the standard text-format output data file.

A.3.1 filename

Define the filename used for text-format data files. This string is expanded as a `strftime` format string and accepts the normal `strftime` format specifiers; for a list of the acceptable format specifiers see `https://docs.python.org/2/library/time.html#time.strftime`. However, since Python expands the string first any percent characters used as part of a `strftime` format specifier must be repeated.

Default: none.

Example:

```
filename = /data/aurorawatchnet/%(site)s/%%Y/%%m/%(site)s_%%Y%%m%%d.txt
```

`%(site)` is replaced with the site abbreviation which was defined previously in the `[DEFAULT]` section. Notice how the `strftime` format specifiers require two % characters.

For June 20th 2014 with the site abbreviation `cwx` this would expand to
`filename = /data/aurorawatchnet/cwx/2014/06/cwx_20140620.txt`

A.4 [awpacket]

Options associated with the standard binary output format. This format is incovenient to read but preserves the received data messages from the magnetometer, and the responses sent back from the recording daemon. It is possible to play back these files to the recording daemon and regenerate other data formats.

A.4.1 filename

See section A.3.1 for a description.

Default: none.

Example:

Typically set to
```
filename = /data/aurorawatchnet/%(site)s/%%Y/%%m/%(site)s_%%Y%%m%%d.awp
```

A.4.2 key

By default the binary data packets are written out with their original signing key. If you plan to make the binary data format available then for security reasons you should probably set a different signing key. You will then be able to share this key without compromising the communication channel with the magnetometer and others will be able to use the error-checking capabilities provide by HMAC-MD5. The key is a 32 character hexadecial string, without any `0x` prefix.

Default: none.

Example:

Typically set to a simple code,
```
key = 00000000000000000000000000000000
```

A.5 [logfile]

Options associated with the recorded log files.

A.5.1 filename

See section A.3.1 for a description.

Default: none.

Example:

Typically set to
```
filename = /data/aurorawatchnet/%(site)s/%%Y/%%m/%(site)s_%%Y%%m%%d.log
```

A.6 [daemon]

Options relating to the data-recording daemon.

A.6.1 connection

Defines the connection between the daemon and the magnetometer. Radio communication emulates a serial port connection. If the magnetometer is PoE model the communication should be set to `ethernet`.

Default: `serial`

A.6.2 close_after_write

Have the daemon close the data and log files after each write. It is recommended that this option only be used on an NFS file system, when used on a file sytem store on flash memory (such as the Raspberry Pi) this may cause excessive writes and early failure of the flash memory. This option may be required for real-time data collection daemons where the data files are also received by `rsync` or `HTTP` uploads.

A.6.3 acknowledge

If set to `false` then acknowledgements are not sent in response to magnetometer data messages.

Default: `true`

A.6.4 read_only

If set to `true` then for serial connections the device is opened read-only and no set up commands are sent.

Read-only mode implies that acknowledgements are not sent, regardless of the setting of the `acknowledge` option.

Default: `false`

A.7 [serial]

Options relating to serial data communication between the magnetometer and data recording daemon. This section is not used when communication uses ethernet.

A.7.1 port

The filename of the serial port. For the Ciseco *Slice of Radio* module this should be set to `/dev/ttyAMA0`, for the Ciseco *URF* module this will probably be `/dev/ttyACM0`.

Default: `/dev/ttyACM0`

A.7.2 baudrate

Baud rate used with the serial port. For the Ciseco *Slice of Radio* module this should be set to `9600`, for the Ciseco *URF* module a higher rate of `57600` can be used.

Default: `9600`

A.7.3 setup

The set-up string which should be sent to the serial device, for defining the channel number used for communication etc..

Default: none.

Typically set to `ATRE;ATCN 25;ATLI R;ATAC`. Ensure that the channel number defined here matches the setting programmed into the magnetometer's EEPROM.

A.8 [ethernet]

These configuration options are used by the data recording daemon only when an ethernet connection with the magnetometer is in use.

A.8.1 local_address

The local IP address used for the data recording daemon. By default the daemon listens on all available interfaces.

Default: empty string.

A.8.2 local_port

The port number of which the recording daemon should listen for magnetometer data packets.

Default: `6588`

A.8.3 remote_address

The IP address of the magnetometer.

A.8.4 remote_port

The port on which the magnetometer expects to receive acknowledgements.

A.9 [controlsocket]

It is possible for the `send_cmd.py` program to send commands to the magnetometer via the data recording daemon. Communication is via a UDP socket or a unix domain socket.

A.9.1 filename

Use a unix domain socket for communication with the data recording daemon, with the given filename. If set to `none` then a control socket will not be created. If the `filename` option is present it takes priority over any `port` option.

Default: none.

A.9.2 port

Use a UDP socket for communication with the data recording daemon, with the given port number. If set to `none` then a control socket will not be created. If the `filename` option is present it takes priority over the `port` option.

Default: `6587`

Example:

```
port = 6587
```

A.10 [magnetometer]

Settings associated with the magneometer.

A.10.1 siteid

The numeric site identifier for the magnetometer. The recording daemon will ignore data packets where the site ID does not match the value set in the configuration file. The site ID should be set as an integer number in the range 0 to 255 inclusive.

Default: none.

A.10.2 key

The HMAC-MD5 key used to sign communication messages sent between the magnetometer and data recording daemon. **This key should be kept secret.** The key is a 32 character hexadecimal string, without any `0x` prefix. The key used here must match that described in Section B.2.7.

Default: none.

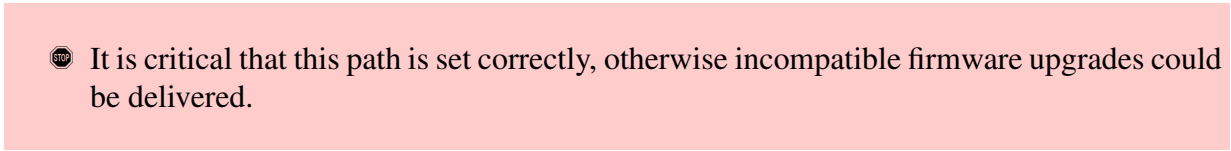
Set to a random value.

A.11 [firmware]

Configuration details for firmware upgrades.

A.11.1 path

The directory in which the firmware upgrades relevant to the magnetometer hardware are stored.



A.12 [upload]

Configuration details for the `upload_data.py` program.

A.12.1 method

Define the upload method in use. Valid options are: `rsync`, `rrsync`, `http`, and `https`. `rsync` uses the `rsync` program to efficiently transfer only the portions of data files which have changed

since the last upoad; `rrsync` is similar but the server restricts which directories may be written to. Both are tunnelled through SSH and thus require the network to permit outward TCP connections to port 22.

The `http` and `https` upload methods make use of the standard HTTP(s) protocol, and therefore require outgoing TCP connections to port 80 and 443 respectively.

A.12.2 rsync_host

Name of the `rsync` host. No facility to set the user is provided, use the SSH `config` file to set the appropriate details.

Default: none.

Example SSH `config` file:

```
Host awn-data
Hostname host.domain.com
User monty
```

A.12.3 url

The URL to which HTTP//HTTPS uploads are sent.

A.12.4 password

The upload password for HTTP and HTTPS methods. Digest authentication is used.

A.12.5 username

The upload username for HTTP and HTTPS methods. If not specified it defaults to the site abbreivation prefixed with `awn-`.

A.12.6 realm

The *realm* used for digest authentication when uploading data with the HTTP or HTTPS method.

A.13 [realtime_transfer]

The data recording daemon is capable of forwarding the incoming binary packets to one or more remote hosts. When this is enabled data is transferred in real-time. The remote host(s) can use the same data recording daemon but should be configured to be read-only so that acknowledgements are not sent.

When data packets are forwarded in this way the daemon does not check that they have been delivered successfully. It is recommended that this method is used in conjunction with the `upload_data.py` program to ensure any data packets which were not delivered are transferred by some other means.

Real-time transfer uses three keys, `remote_host`, `remote_port` and `remote_key`. To send data to multiple hosts a suffix is applied to each of this keys to group the settings together, e.g., `remote_host2`, `remote_port2` and `remote_key2`. The suffix must not contain whitespace characters.

A.13.1 remote_host

The hostname or IP address to which data packets are sent.

A.13.2 remote_port

The UDP port to which data packets are sent.

A.13.3 remote_key

The HMAC-MD5 key used to sign the data packets. This should be different to the key used to communicate with the magnetometer.

A.14 [dataqualitymonitor]

The data recording daemon can monitor for the existence of a file indicating that data quality may be compromised (e.g., due to local site activities such as cutting grass). If the file is present an extension is appended to the data files to clearly separate the poor quality data. Any real time data transfers in operation are suspended. When the file is removed the normal filenames are used again and real time data transfer resume as normal.

A.14.1 port

Defines the serial port used by the awnetd_monitor.py daemon. If using a USB device it is recommended to modify the udev rules so that a fixed device name is generated based on the serial number.

Example udev rule to create a symbolic link for an FTDI USB serial adaptor, save to /etc/udev/rules.d/90-usb_serial.rules or similar:

```
# Have symlinks based on serial number for FTDI devices. Used for
# awnetd_monitor
SUBSYSTEMS=="usb", KERNEL=="ttyUSB[0-9]*", \
ATTRS{idVendor}=="0403", ATTRS{idProduct}=="6001", SYMLINK+="tty_ftdi_%s{serial}"
```

Default: none.

A.14.2 filename

The name of the file which when present indicates reduced data quality. This file can be created by the awnetd_monitor daemon or an external process.

Default: /var/aurorawatchnet/data_quality_warning

A.14.3 extension

The extension which is appended to files to indicate reduced data quality.

Default: .bad

A.14.4 pidfile

A file containing the process ID of the awnetd_monitor daemon.

Default: none.

A.14.5 logfile

The file where the log output of the awnetd_monitor daemon is written.

Default: none.

A.14.6 led_active_low

A flag indicating if the DTR signal line should be pulled low to turn on the LED.

Default: true.

A.15 [ntp_status]

Configuration details for data recording daemon (`awnetd.py`) and also used by the `check_ntp_status` program. As the Raspberry Pi does not have an onboard real-time clock it must obtain the time from the network using NTP. At boot time, and other occasions when the NTP servers are not accessible, the system clock on the Raspberry Pi may not be correct. The data recording daemon can be made aware that NTP is running and correctly synchronized by the presence of a semaphore file. The file's age is also checked to ensure that the presence of a stale semaphore file is not misinterpreted. When the data recording daemon is configured to check NTP status the current time is sent to the microcontroller only if NTP is synchronized.

A.15.1 filename

The filename of the semaphore file. Required for `check_ntp_status`. Not required by `awnetd.py` and if missing NTP status is not checked.

A.15.2 max_age

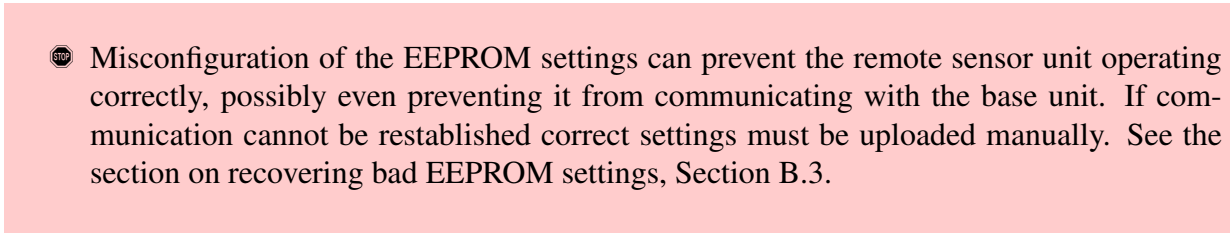
The maximum allowable age in seconds for the semaphore file to be considered valid. The presence of a file older than this age is ignored. Required for `check_ntp_status`. Not required by `awnetd.py` and if missing NTP status is not checked.

Appendix B

EEPROM settings

B.1 Introduction

The EEPROM settings determine the default behaviour of the remote sensor unit, controlling various features such as the sampling interval and site ID. The settings are also select important features such as the radio or ethernet controller.



Misconfiguration of the EEPROM settings can prevent the remote sensor unit operating correctly, possibly even preventing it from communicating with the base unit. If communication cannot be restablished correct settings must be uploaded manually. See the section on recovering bad EEPROM settings, Section B.3.

B.2 Settings

The settings are described below using the Python option name that is passed to the `generate_eeprom_settings.py` and `send_cmd.py` programs.

B.2.1 `-eeprom-adc-address-list`

A comma-separated list of the 3 I2C addresses for the magnetometer sensor ADC(s). Addresses may be given in hexadecimal format if preceded with `0x`, otherwise decimal is assumed. Missing ADCs should be indicated by giving the value 255.

Example:

```
-eeprom-adc-address-list 0x6E,0x6A,0x6C
```

B.2.2 -eeprom-adc-channel-list

A comma-separated list of the 3 I2C channels for the magnetometer sensor ADC(s). Unless there is a good reason to swap the ADCs all channels should be 1.

Example:

```
-eeprom-adc-channel-list 1,1,1
```

B.2.3 -eeprom-adc-gain-list

A comma-separated list of the 3 I2C gains for the magnetometer sensor ADC(s). Valid values for the gain are 1, 2, 4, and 8. If the gain is set too high there is a possibility that the output value will saturate. With the standard 100 000 nT range FLC100 sensors the gain should normally be set to 1 or 2. With HEZ alignment the E channel may be set as high as 8.

Example:

```
-eeprom-adc-channel-list 1,8,1
```

B.2.4 -eeprom-aggregate 0,1,2

Select the method used to compute the sample value when oversampling is in operation. Valid values are:

0 Mean.

1 Median.

2 Trimmed mean.

B.2.5 -eeprom-all-samples 0,1

When oversampling is in operation have all data samples returned. Note that this can cause the messages to become too large to send correctly when a large number of oversamples are taken, particularly with a 3 axis magnetometer system.

B.2.6 -eeprom-comms-type 0,1,2

The installed firmware may support multiple methods of communication between the remote sensor unit and the base station. If so this option indicates which method should be used. It is vital that this setting is correct otherwise the remote sensor unit will not function until reprogrammed. Valid values are:

0 XRF radio.

1 RFM12B radio.

2 W5100 Ethernet, using UDP packets.

Note that automatic selection may not work correctly in all cases and is included as an option of last resort (e.g. when encountering an unprogrammed EEPROM).

B.2.7 `-eeprom-hmac-key EEPROM_HMAC_KEY`

The HMAC-MD5 key used to sign communication messages sent between the magnetometer and data recording daemon. **This key should be kept secret.** The key is a sequence of 16 numbers between 0 and 255 (inclusive). Values may be given in hexadecimal format if preceded with `0x`, otherwise decimal is assumed. The key used here must match that described in Section A.10.2.

B.2.8 `-eeprom-num-samples NUMBER`

Enable oversampling; take NUMBER samples and compute an aggregate value. See also sections sec:eeprom-aggregate and sec:eeprom-all-samples.

B.2.9 `-eeprom-sampling-interval-16th-s DURATION`

The initial sampling interval, defined in units of $\frac{1}{16}$s. The `send_cmd.py` command, when used with its `-sampling-interval` option, can be used to modify sampling interval without requiring a reboot.

B.2.10 `-eeprom-site-id EEPROM_SITE_ID`

Set the unique site identifier number.

B.3 Recovering bad EEPROM settings

The easiest method to recover from bad EEPROM settings is to generate new settings using the `generate_eeprom_image.py` program. The original settings used at manufacture should only be used if the firmware has not been updated, otherwise the settings may be missing values required by the newer firmware. Further information on generating and uploading new EEPROM settings can be found in Section 4.4.3.