DACCIWA Rain Gauge Manual: v1.0

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Principle of Operation

The sensor operates on the principle that rain falling into the collector of the gauge is funnelled through an orifice of known diameter resulting in the formation of a drop with the same diameter as the orifice. Knowing the cross-sectional area of the collector and the diameter of the orifice a single drop from the orifice can be expressed in terms of a depth of rain fall. For this sensor:

1 drop is equivalent to 0.01 mm

Drops issuing from the orifice fall past an IR (infra red) sensor resulting in the generation of a 12V amplitude TTL pulse. The maximum measurable rainfall rate is limited either by the ability of the logging system to distinguish individual droplets (TTl pulses) or by the process of drop formation. Assuming the logger can distinguish between individual drops (TTL pulses) the maximum rain rate is determined by the point at which individual droplets issuing from the orifice merge to form a stream: this occurs for rain fall rates of ~300 mm/hr.

The logger, details of which are in the following sections, uses a fast sampling interrupt to count the number of drop induced TTL pulses occurring per minute. The interrupt is triggered on a rising edge thus as the rain fall rate increases the count reported increases, however, this will drop to zero if the rain fall rate is such that a continuous stream emerges from the orifice.

Note: Zero reported droplets can occur either due to no droplets forming (it is not raining or the collector is blocked) or its raining too much (> 300mm/hr), determining which scenario applies requires inspection of the rainfall history.

The Logger

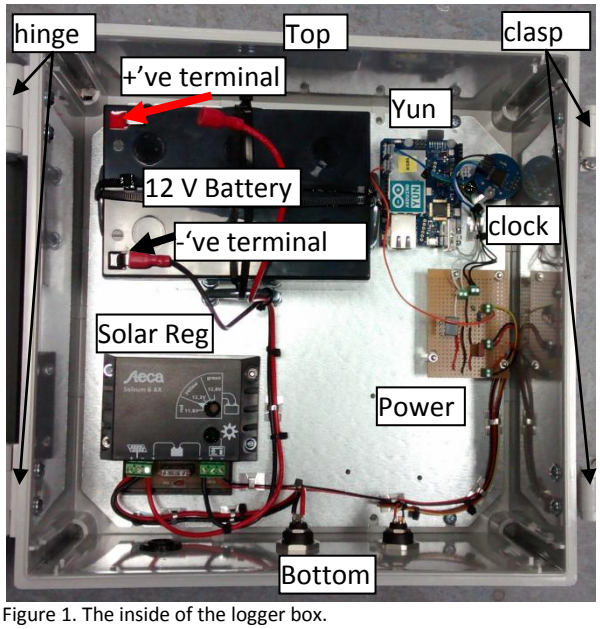


Figure 1. The inside of the logger box.

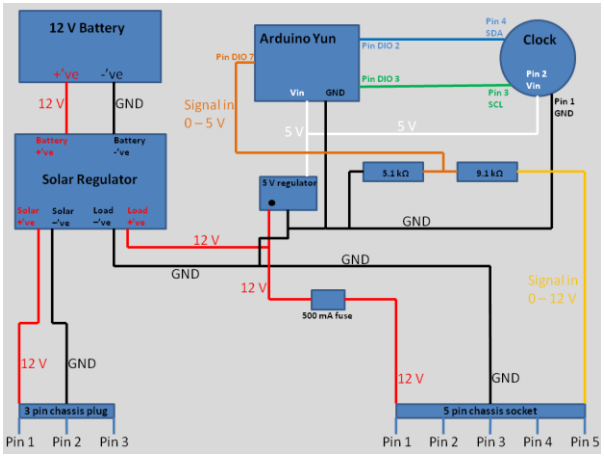


Figure 2. Schematic of logger showing circuit and pin assignment.

Figure 1 Shows a photograph of the internal arrangement of the logger box and figure 2 a schematic of the circuit.

The logger is based around the Arduino Yun microcontroller: a microcontroller based on the ATmega32u4 and the Atheros AR9331 chips. The Atheros processor supports a Linux distribution based on OpenWrt named OpenWrt-Yun. The board has built-in Ethernet and WiFi support, a USB-A port, micro-SD card slot, 20 digital input/output pins, a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a 3 reset buttons. The Atheros AR9331 is used purely to control external access to the system and data files while the actual logging in performed by ATmega32u4.

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Although there is an inherent system clock built into the linux distribution this is neither used nor set in this application: ALL MEASUREMNT TIMINGS ARE DERIVED FROM THE REALTIME CLOCK (RTC).

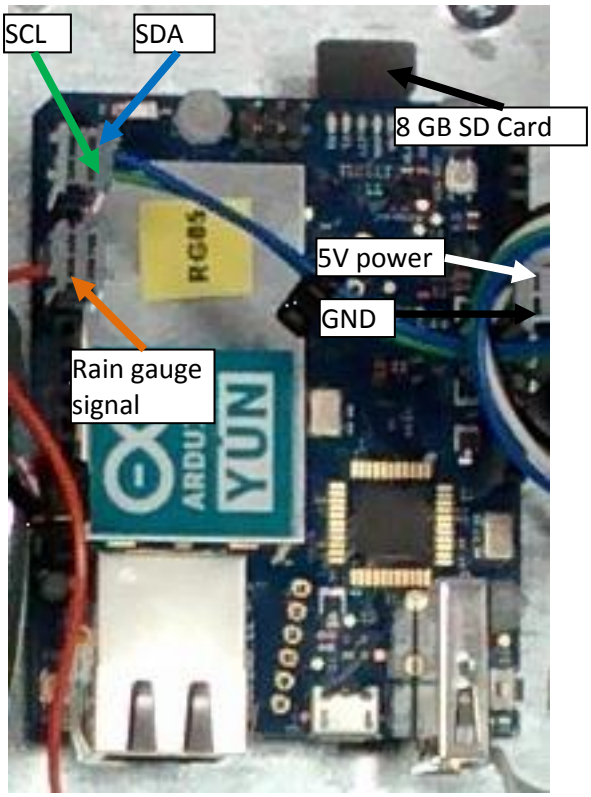


Figure 3: The Yun microcontroller

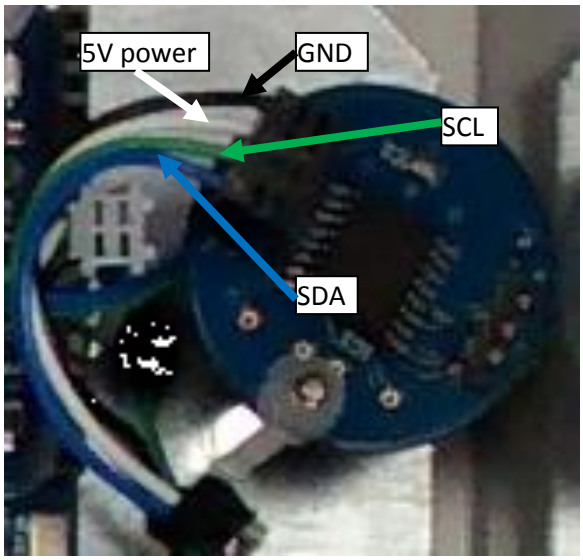


Figure 4. The real time clock

Measurement timing is derived from the battery backed, temperature compensated RTC. When initialised at the time of logger manufacture this clock is set from the system clock of the host PC being used as the programming interface. This process takes a finite amount of Yun processing time resulting in an approximate 90 second offset of the logger from the host PC clock (logger lags). Further offsets from UTC can occur if the host PC is not regularly synchronised with a known time standard. In this case the host PC is synchronised hourly, hence the only offset is that arising from the processing time.

The clock is battery backed and the battery should last ~8 years. The battery backup ensures the correct time is kept even if the there is no power from the solar panel or battery. The time slip on the RTC is < 0.0864 seconds per day.

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

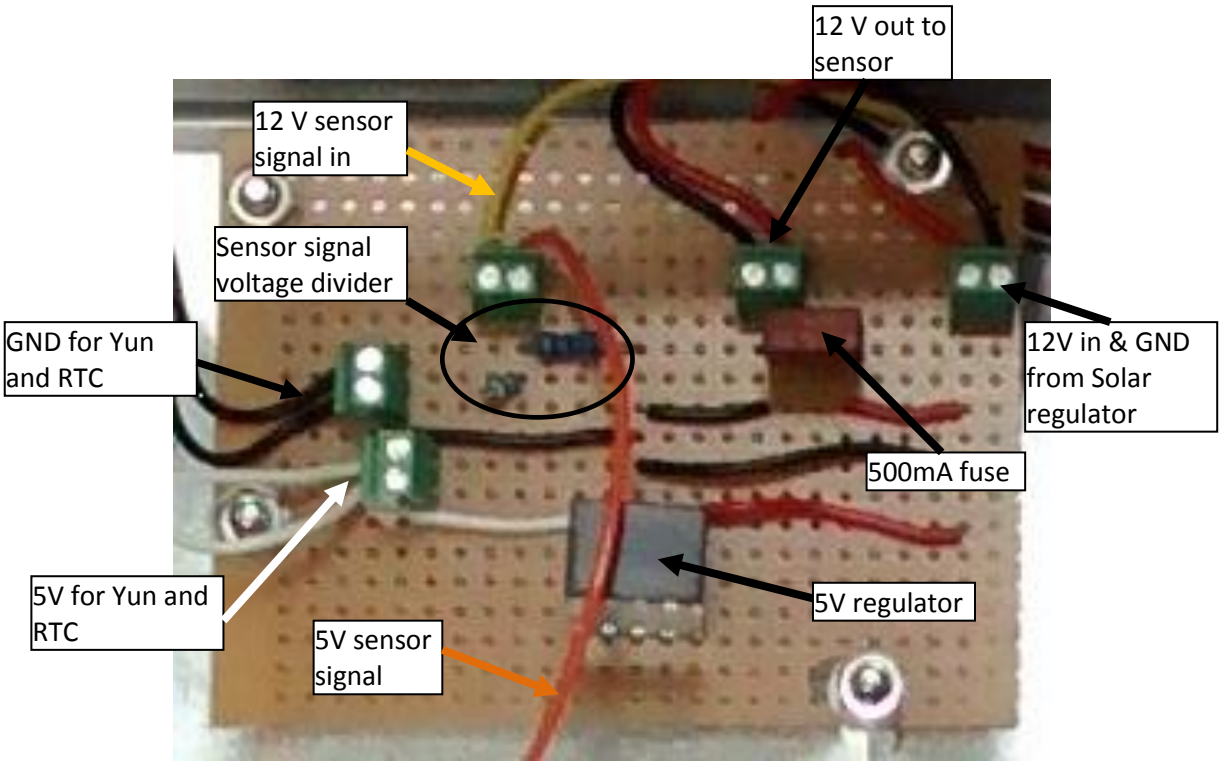


Figure 5: Power distribution and signal processing .

Power and sensor signal is distributed via the distribution board shown in figure 5. 12V from the solar regulator (figure 7) is connected to the distribution board via the screw terminals as shown. This power is directed to both a 5V voltage regulator and via a 500ma fuse to the rain sensor (via the screw terminals shown). Both the Yun and RTC have no inbuilt voltage regulation and direct application of 12V to these the boards would result in their damage. The 5V voltage regulator ensures that 5V, and 5V only, is directed to the Yun and RTC boards and connection is made via the screw terminals as shown.

As discussed earlier when a droplet passes the IR sensor in the gauge a 12V pulse is developed. The maximum voltage input to the Yun signal inputs is 5V and damage to the board would occur if the sensor was connected directly to the micro-processor. To prevent damage the 12 V raw sensor signal is conditioned to produce a 4.3V signal. The conditioning is performed using a simple voltage divider (hi precision resistor of value 9.1kΩ and 5.1kΩ) and this is ringed in figure 5.

The data logger and sensor are powered by a combination of solar panel and battery. The solar regulator shown in figure 6 controls the voltage from the solar panel and ensures that the battery (figure 7) is charged correctly. This regulator determines which of the power sources (panel or battery) is the greater (maximum if 13V) and directs this to the load. Switching between sources is transparent and does not affect the behaviour of the system. The regulator also charges the battery and dissipates excess energy as heat when required.

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Figure 6: Solar regulator

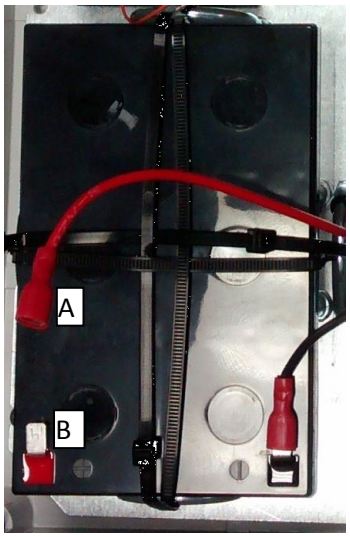


Figure 7: Battery

Initial connection

1. Check all wires except the red to the battery are connected
    a. Should any other wires be unconnected consult wiring diagram and reconnect.
2. Connect battery – red wire (A in figure 7) to red battery spade (B in figure 7)
    a. Lights on Yun should flash
    b. System will automatically start logging – no need to do anything more
3. Connect solar panel
    a. Solar regulator has a green LED and an LED that can change colour
    b. Green LED is on as soon there is energy from the solar panel
    c. When the regulator starts to limit current to the battery (the charge current) this LED will flash
    d. The LED which can change colour shows the battery voltage by its colour

Data

The data is logged to the 8Gb SD card on the Yun. Files are ASCII with comma separated variables (standard .csv format) and the file naming protocol is:

<unit ID>_<YYMMDDHHmmSS>.csv

Files are created automatically when the system is powered and at the start of a new day. Consider the file RG05_141101143754.csv: this file was created by logger RG05 at 14:37:54 on 1st November 2014

The data on the SD card can either be access by following the procedure below:

1. Unplug the solar panel
2. Disconnect the battery
3. Remove the SD card

which is not recommended or by using the Yun WIFI capability (recommended). How this is done is covered in the following paragraph.

The Yun should appear as a wireless network connection with a name:

Arduino Yun-XXXXXXXXXXXX.

Connect to this network (you should not be asked for any login or password at this stage). The Yun will have an IP address of:

192.168.240.1

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

on this network and a user can log in to it using any terminal program (for example Putty: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html).

- Login as : root
- Password: DACCIWARG

The data path is /mnt/sda1 and data can be copied off using the SCP protocol. From windows using WinSCP is recommended:

- Login as : root
- Password: DACCIWARG
- Set protocol to SCP

Note the linux system time is not set by the logger. The RTC derived time used to time stamp data is independent of the linux system time and the linux system time is not used in any way in the operation of the logger. You should expect to see a difference between the displayed file modification time and the true time as reported in the data files.

Data consists of 7 comma separated variables, these are formatted as follows:

- Column 1 = year – 2000
- Column 2 = month
- Column 3 = day
- Column 4 = hour
- Column 5 = minute
- Column 6 = second
- Column 7 = number of drops counted in last minute

and a random example of data is shown below. Note that this is minute averaged data so the seconds (column 6) is always zero

14,	11,	14,	11,	59	0,	0
14,	11,	14,	12,	0,	0,	0
14,	11,	14,	12,	1,	0,	0
14,	11,	14,	12,	2,	0,	0
14,	11,	14,	12,	3,	0,	1
14,	11,	14,	12,	4,	0,	1
14,	11,	14,	12,	6,	0,	3
14,	11,	14,	12,	8,	0,	4
14,	11,	14,	12,	9,	0,	5
14,	11,	14,	12,	10,	0,	1
14,	11,	14,	12,	11,	0,	1
14,	11,	14,	12,	12,	0,	0
14,	11,	14,	12,	12,	0,	0

Mounting Sensor

The Rain Gauge sensor itself requires appropriate mounting in order to operate correctly. The sensor has 10m of cable attached and should be mounted following the instructions below. Note that a drainage hole is already present and there should be no need to open the bowl of the gauge.

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Figure 8a: Rain gauge sensor - top

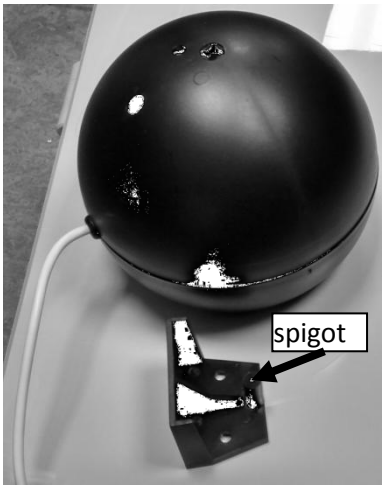


Figure 8b: Rain gauge sensor – bottom. Mount removed

Figure 8a and b show the rain sensor. The spigot of the mount in figure 8b simply fits in to the hole on the underside of the sensor. This mount can be attracted to a wall or post with bolts or screws.

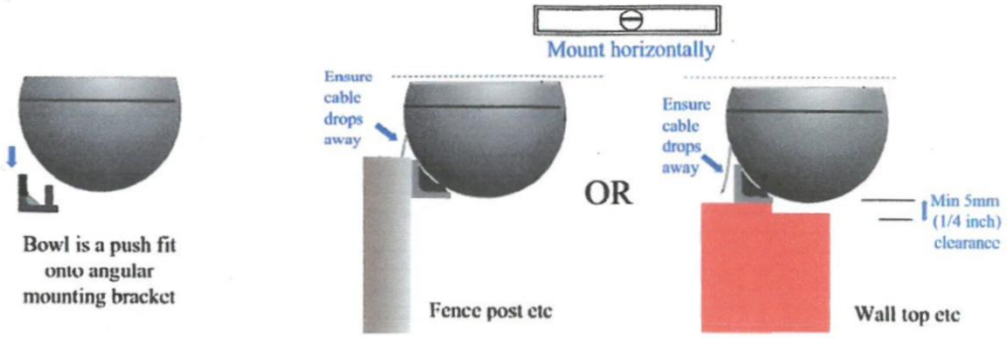
Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

STAND ALONE RAIN SENSOR INSTALLATION INSTRUCTIONS

Siting

The Rain Sensor should, ideally, be situated at a distance of twice the height of local obstructions and mounted 30cm (12") above ground level. This requirement can often be difficult or impossible to achieve due to surrounding bushes, trees etc. For most practical purposes accuracy will not be greatly impaired if the site is only a distance equal to the height of the obstructions. It is important that the sensor is reasonably accessible as, from time to time, the filter will require cleaning. Very often the top of a fence post provides an excellent site so long as it is firm, clear of bushes and trees as above. It is wise to avoid mounting on or near buildings which can cause wind turbulence and create dust. Avoid also mounting near to transmitter aerials which can in certain circumstances cause interference.

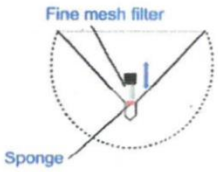
Installation



Screw the angular mounting bracket to a ***firm and rigid*** post, fence or wall ensuring that the funnel opening is absolutely horizontal. Ensure also there is at least ***5mm clearance*** between the ***bottom of the bowl and any surface below.*** To avoid the possibility of water running down the cable into the sensor, it is important that the cable drops away from the bowl. Leave some slack in the cable beneath the sensor to facilitate cleaning. Ensure that the interface cable is not in close proximity to power and transmitter cables. Be very careful not to nick the cable when being clipped to walls etc. If the cable is to be buried, run cable through plastic hose etc. to avoid vermin biting through cable.

Maintenance

From time to time dust and other foreign bodies will accumulate on the funnel filters.
To clean;- carefully withdraw filter and sponge and clean under a tap.
Important - avoid touching the stainless steel filter mesh with bare fingers as this will deposit grease which may impair the flow of water.



Mounting the solar panel

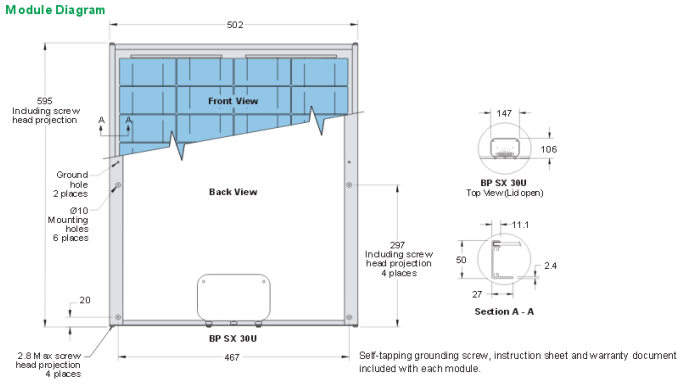


Figure 9: Solar Panel

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

The solar panel, a schematic of which is shown in figure 9, has an aluminium frame surrounding in. It can be seen that there are number of regularly spaced pre-drilled holes on this frame that can be used to mount the panel.

When mounting a panel care must be taken in its positioning both form the point of view of the site and aspect. The user is recommended to visit:

http://solarelectricityhandbook.com/solar-angle-calculator.html

to determine the best installation angle for the panel. It is also recommended that the panels be wiped clean of dust and other foreign bodies at regular intervals; this will improve collection efficiency.

The panel cable length is approximately 2m

Mounting the logger box

The logger box should be mounted vertically such that the connectors for the sensor are to the bottom and the door hinge is on the left when looking from the front. Figure 10 shows the dimensions of the logger box and the position of the mounting holes. These holes are positioned at the corners on the underside of the box and have tapped inserts that allow M6 machine bolts to be used to attach the enclosure to a wall or mounting frame.

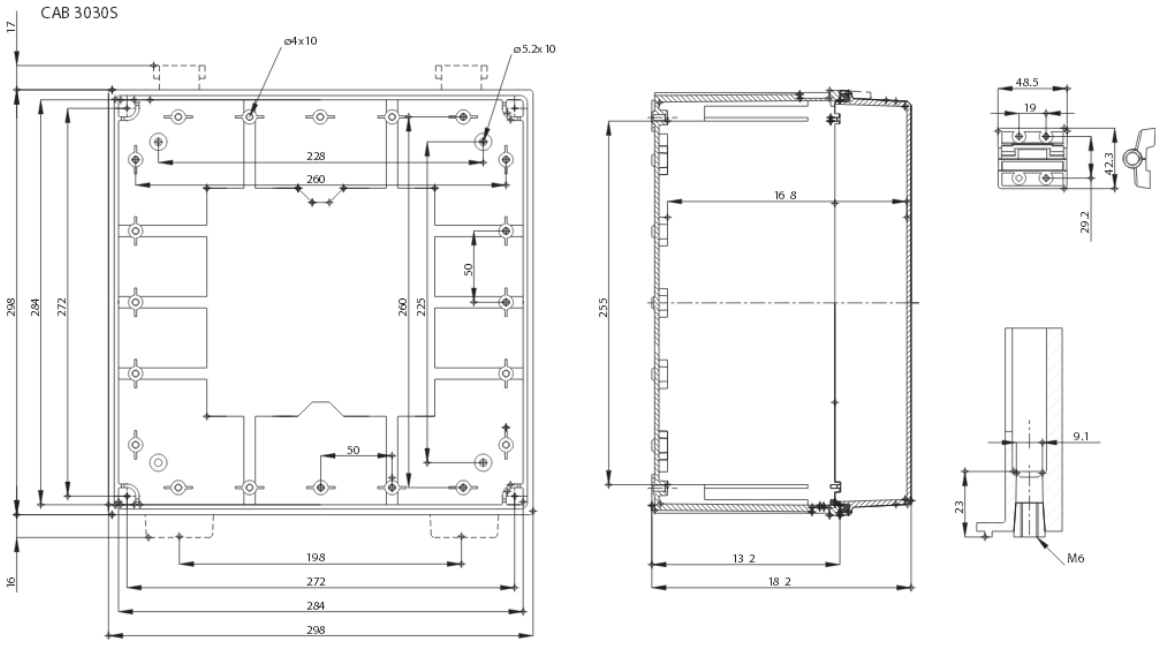


Figure 10: Schematic of the logger box

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Appendix 1: Pin outs.

1. Solar Panel
   3 pin cable socket (Binder 720 series: 99 9106 00 03)
   - Pin 1: +Vcc brown
   - Pin 2: Gnd blue
   - Pin 3: NC

2. Rain Gauge
   5 pin cable plug (Binder 720 series: 99 9113 03 05)
   - Pin 1: +Vcc red
   - Pin 2: NC
   - Pin 3: Gnd black
   - Pin 4: NC
   - Pin 5: signal yellow

3. Logger
   3 pin chassis plug (Binder 720 series: 99 9107 00 03)
   - Pin 1: +Vcc red
   - Pin 2: Gnd black
   - Pin 3: NC

   5 pin chassis socket Binder 720 series: 99 9116 00 05)
   - Pin 1: +Vcc red
   - Pin 2: NC
   - Pin 3: Gnd black
   - Pin 4: NC
   - Pin 5: signal yellow

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Appendix 2: Yun pin assignment

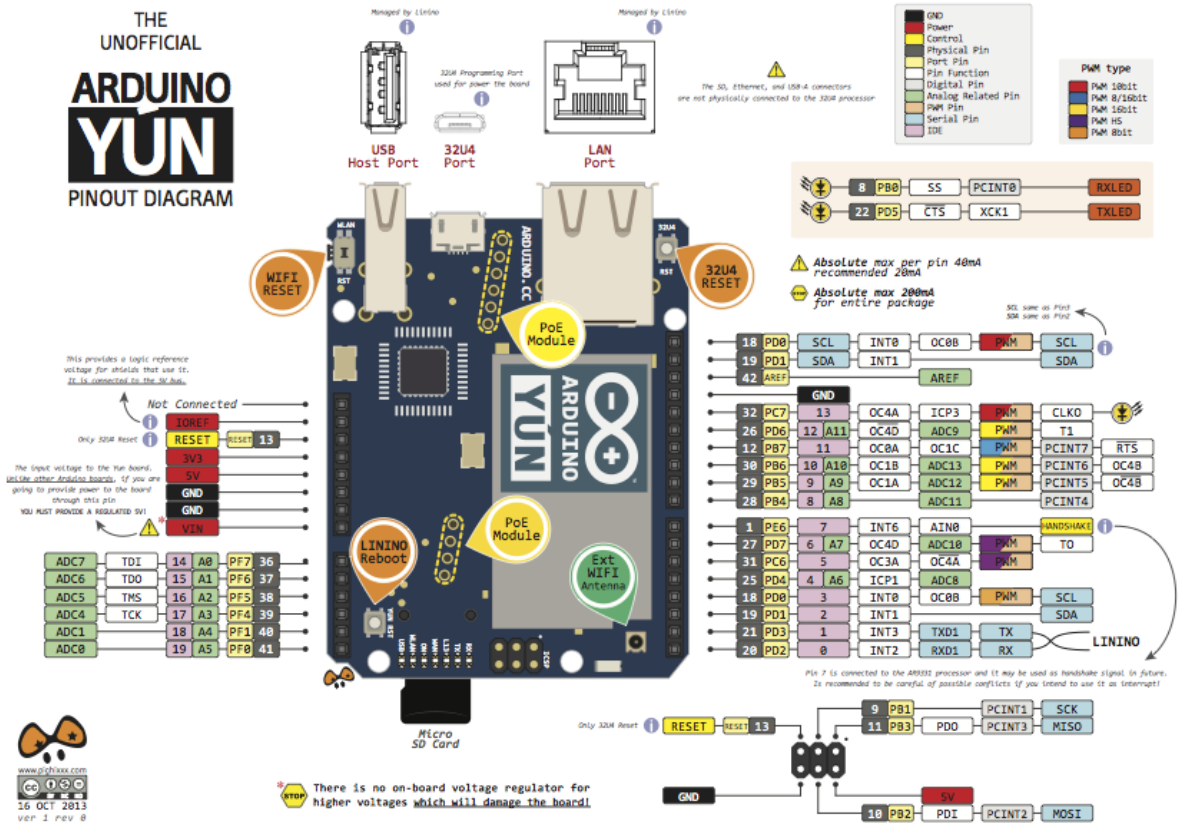


Figure A2.2: Arduino Yun pin designation.

Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

Appendix 3: Yun Sketches.

To program the Yun using these sketches the Arduino IDE will be required (http://arduino.cc/en/main/software). Note the password DACCIWARG will be required to upload a sketch to a logger and the installation of the following additional RTClib.h library (available from: https://github.com/adafruit/RTClib) is required.

1. Set Real Time Clock

```
#include <Bridge.h>
#include <Console.h>
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 RTC;

void setup () {
   Bridge.begin();          // Initialize the Bridge
   Console.begin();
   Wire.begin();
   RTC.begin();
   RTC.adjust(DateTime(__DATE__, __TIME__));
}

void loop () {
   DateTime now = RTC.now();

   Console.print(now.year(), DEC);
   Console.print('/');
   Console.print(now.month(), DEC);
   Console.print('/');
   Console.print(now.day(), DEC);
   Console.print(' ');
   Console.print(now.hour(), DEC);
   Console.print(':');
   Console.print(now.minute(), DEC);
   Console.print(':');
   Console.print(now.second(), DEC);
   Console.println();

   delay(1000);
}
```


Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

2. Check Real Time Clock

```
#include <Bridge.h>
#include <Console.h>
#include <Wire.h>
#include "RTClib.h"

RTC_DS1307 RTC;

void setup () {
   Bridge.begin();          // Initialize the Bridge
   Console.begin();
   Wire.begin();
   RTC.begin();
}

void loop () {
   DateTime now = RTC.now();

   Console.print(now.year(), DEC);
   Console.print('/');
   Console.print(now.month(), DEC);
   Console.print('/');
   Console.print(now.day(), DEC);
   Console.print(' ');
   Console.print(now.hour(), DEC);
   Console.print(':');
   Console.print(now.minute(), DEC);
   Console.print(':');
   Console.print(now.second(), DEC);
   Console.println();

   delay(1000);
}
```


Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

3. Logging

```
#include <Bridge.h>
#include <Console.h>
#include <Wire.h>
#include <Process.h>
#include <FileIO.h>
#include "RTClib.h"

RTC_DS1307 RTC; //create occurrence of clock
String hostname="RG01"; // change for each unit
volatile int pulseCount; // hardware interrupt counts
byte sensorInterrupt = 4; // 3 (interrupt 0), 2 (interrupt 1), 0 (interrupt 2), 1 (interrupt 3)
and 7 (interrupt 4).
int currentday, currentminute;
byte dt[6]; // output date time array
int drops;
DateTime now;
File DataFile;
char fn[70];

void make_file() {
  String xx;
  String yy =  String(now.year()-2000, DEC);
  String mm =  String(now.month(), DEC);
  String dd =  String(now.day(), DEC);
  String hh =  String(now.hour(), DEC);
  String MM =  String(now.minute(), DEC);
  String ss =  String(now.second(), DEC);

  if (yy.length()!=2) yy="0"+yy;
  if (mm.length()!=2) mm="0"+mm;
  if (dd.length()!=2) dd="0"+dd;
  if (hh.length()!=2) hh="0"+hh;
  if (MM.length()!=2) MM="0"+MM;
  if (ss.length()!=2) ss="0"+ss;

  xx="/mnt/sda1/" + hostname + "_" + yy + mm + dd + hh + MM + ss + ".csv";
  xx.toCharArray(fn,50);
  File DataFile = FileSystem.open(fn, FILE_WRITE);
  DataFile.close();
}

void setup() {
  Bridge.begin();           // Initialize the Bridge
  Console.begin();
  FileSystem.begin();
  Wire.begin();
  RTC.begin();
  attachInterrupt(sensorInterrupt, pulseCounter, RISING);
  delay(100);
```


Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk

```
  now = RTC.now();
  make_file();
  currentday=now.day();

  //wait for start of new minute
  do {
    now = RTC.now();
    delay(500);
  } while (now.second()!=0);
  currentminute=now.minute();
  pulseCount=0;
}

void pulseCounter() {
  pulseCount++; // Increment the pulse counter
}

void prep_data() {
  dt[0]=now.year()-2000; dt[1]=now.month(); dt[2]=now.day();
  dt[3]=now.hour(); dt[4]=now.minute(); dt[5]=now.second();
}

void save_data() {
  File DataFile = FileSystem.open(fn, FILE_APPEND);
  DataFile.print(dt[0],DEC); DataFile.print(","); DataFile.print(dt[1],DEC); DataFile.print(",");
  DataFile.print(dt[2],DEC); DataFile.print(","); DataFile.print(dt[3],DEC); DataFile.print(",");
  DataFile.print(dt[4],DEC); DataFile.print(","); DataFile.print(dt[5],DEC); DataFile.print(",");
  DataFile.println(drops);
  DataFile.close();
}

void loop() {
  now = RTC.now();

  if (now.minute()!=currentminute){ // 1 minute average
    drops=pulseCount; pulseCount=0;
    currentminute=now.minute();
    if (now.day()!=currentday){
      currentday=now.day();
      make_file();
    }

    prep_data();
    save_data();
  }
}
```


Dr Barbra Brooks, NCAS, School of Earth and Environment, University of Leeds, Leeds. LS2 9JT. UK
barbara.brooks@ncas.ac.uk