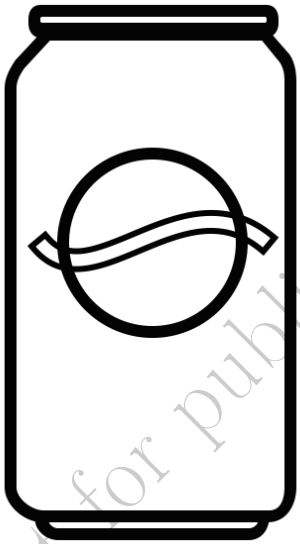
qbcan User Manual

22nd August 2015



Contents

Cover logo created by Claire Jones from the Noun Project

Part I
System description

qbcan is a versatile, easy-to-use and high performance CanSat bus. The qbcan bus is provides all the required capabilities of a minimalistic CanSat: communications, a temperature and pressure sensor, and a computing platform with wide range of interfaces. The user can then add extra functionality (adding sensors and actuators via the provided interfaces) and develop more complex The main components of qbcan are:

- Arduino Pro Micro microcontroller.
- RFM69HW 433MHz transceiver.
- BMP180 temperature and pressure sensor.
- Software library that interfaces with the transceiver and the pressure and temperature sensor for rapid development.
- 9V Battery.
- 2 M2 mounting holes.

The small qbcan footprint, the exposed Arduino Pro Micro pins and its mounting holes allow qbcan to be flexible with respect its mounting location, allowing CanSat developers to mount it wherever is more convenient.

qbcan (hardware and software) is open-source and the users are encouraged to modify it and re-distribute their work.

1 Microcontroller

The core of qbcan is an Arduino Pro Micro microcontroller. The microcontroller provides the required computing power to the CanSat. The wide range of interfaces provided by the the Arduino Pro Micro allows the user to develop more complex CanSat mission by integrating additional peripherals. The main features of the Arduino Pro Micro are:

- ATmega32U4 running at 5V/16MHz.
- Easy to program using the Arduino Integrated development environment.
- On-Board micro-USB connector for programming.
- I2C, SPI and UART serial communication ports.
- 4 channels to read analogue signals using a 10-bit analogue to digital converter.
- 5 Pulse Width Modulated output pins.
- 12 Digital Input Output pins.
- Tiny footprint 33×17.8 mm.

The pinout of the Arduino Pro Micro can be seen in Fig. 1. All these pins and communication interfaces are available to the user except pin 7 and 10 which are used to communicate by the transceiver (the SPI and I2C interfaces are shared with the transceiver and the pressure and temperature sensor but are also available to the user).

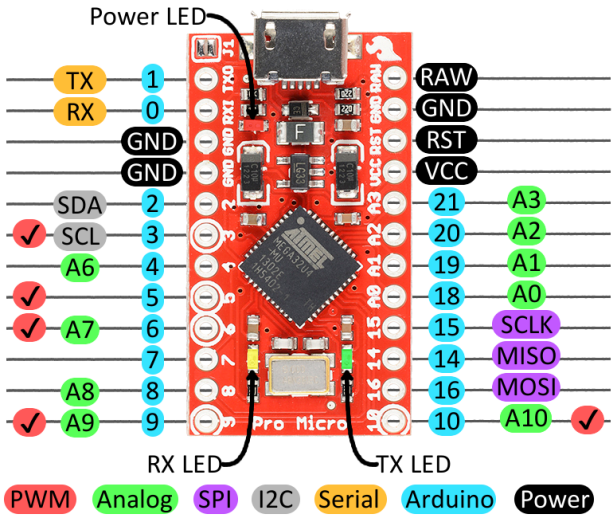


Figure 1: Arduino Pro Micro pinout.

2 Transceiver

A RFM69HW 433 MHz transceiver is included to provide long range communications capabilities to qbcan. The main features of the transceiver are:

- +20 dBm - 100 mW power output capability.
- High sensitivity: down to -120 dBm at 1.2 kbps.
- Programmable output power: -18 to +20 dBm in 1dB steps.
- Fully integrated synthesiser with a resolution of 61 Hz.
- Frequency selectable by software over 256 different channels.
- 255 possible nodes in every channel.
- FSK, GFSK, MSK, GMSK and OOK modulations.
- Hardware 128bit AES encryption.
- Over 400+ meters range using whip antennas and several km range using a Yagi antenna on the receiving end.

The transceiver communicates with the Arduino over the SPI interface and uses the pin 10 of the Arduino as the Slave Select (that is why this pin is reserved). The transceiver software, included with qbcan, is interrupt driven (asynchronous response to incoming communications), using the Arduino pin 7 to provide this interrupt and thus pin 7 is also not available to the user. The antenna of the transceiver is a simple quarter wavelength monopole antenna.

Using this transceiver a qbacn can be used a CanSat while another qbcan can be used as a "ground station", receiving telemetry from the CanSat (and sending to a PC via the USB port) and sending commands to the CanSat.

3 Temperature and pressure sensor

qbcan includes a BMP180 barometric pressure and temperature sensor. This sensor communicates over I2C and provides:

- Pressure sensing range: 300-1100 hPa (9000m to -500m above sea level).
- Up to 0.02hPa / 0.17m altitude resolution.
- -40 to +85°C operational range, +-2°C temperature accuracy.

4 Power

qbcan is powered by an included 9V battery. The connector to the battery is included and by using the available Arduino pins the user is able to access the raw 9V from the battery - from the Arduino Raw pin - and 5V regulated power - from the Arduino VCC pin [1] . If you plan to use more than 500 mA from the 5V line it is recommended that you use the raw battery voltage and use your own voltage regulator.

5 Library

A software library is included providing an easy-to-use interface with the transceiver and pressure sensor. Example code is provided to speed up development of the CanSat mission. The library includes a CanSat example and a "ground station" example.

[1] A 3.3V output is planned for the final qbcan version.

Part II
Getting started

The qbcan bus comes as a kit and needs to be assembled. Also to make develop software for the Arduino Pro Micro and use the provided library the development computer needs to be configured appropriately. Section 1 covers the physical assembly of qbcan and Sec. 2 provides a step by step guide to get the development environment up and running.

To complete the software installation you will require an assembled qbcan but steps 2-2.3 only require the Arduino Pro Micro on its own so you can do them before starting the assembly.

1 Assembly

The assembly of qbcan is simple and only requires basic soldering skills. If you don't have any experience soldering then it will probably take you around 2 hours (less than 1 hour if you have soldering experience). Also, the tools required to assemble qbcan are those required for soldering hobby electronics. We recommend you to have the following equipment:

- Pliers and/or tweezers.
- Soldering iron and solder. It is important that the solder comes with a flux core (this will facilitate soldering).
- Optional but recommended - Flux remover (to clean flux residue). It is important to use a remover for you specific flux formulation. Some flux formulations do not require to clean the residue afterwards. If you don't have a specific flux remover at hand, Isopropyl alcohol or acetone could help remove most of the residue.
- Optional - A "third hand" (see Fig. 2) it is very convenient when you are alone when doing the assembly.
- Optional - Solder Wick remover in case you make any mistake and want to desolder some components.
- Optional - Extra flux (e.g. in a pen dispenser). This element is optional but can help solder the voltage regulator (that is surface mounted).

If you don't have any experience with soldering hobby electronic components we recommend you to watch the excellent EEVblog soldering tutorial. Part 1 covers the equipment, Part 2 covers through hole soldering techniques and Part 3 covers surface mount soldering (required to solder the voltage regulator).

To assemble the qbcan board you just need to follow these steps:

1.1 Clean the board

With a pair of pliers remove any imperfection from the board (bits leftover from the fabrication process)[2].

1.2 Solder the headers onto the Logic Level Converter and the BMP180.

The first soldering will be an easy one. We will be soldering the headers onto the Logic Level converter and the BMP180. Use the short 0.1" (2.54 mm) pitch headers for this (it does not mater in which orientation you solder the headers). Make sure that the headers are straight and not at an angle with the board. The final result for the Logic Level Converter is shown in Fig. 4.

[2]This imperfections where specific to the fabrication process used for the prototype so in the final product this may not exist.

Figure 2: Example of a what is known as a third hand.

As we will be using the same headers for the BMP180 but the BMP180 board has one connector less (5 instead of 6) we need, with the help of the pliers, to break away one of the pins from the headers (see Fig. 5).

When that is done you can solder the header into the BMP180 board. The final result is shown in Fig. 6.

When that is inspect the solder and make sure that you have good quality connections, also before continuing remove any flux residue.

1.3 Solder Arduino Pro Micro headers.

Now that you have some practice in soldering headers we will solder the headers of the Arduino Pro Micro. The procedure is exactly the same but using the long 0.1" (2.54 mm) picth headers. Leave the long pins on the top side of the Arduino. The final results should look like Fig. 7.

The long pins will enable your peripherals (GPS, actuators or other sensors) to connect to any pins of the Arduino by using female headers or female jumper wires (see Fig. 8).

Again, before continuing inspect the connections and clean any flux residue.

1.4 Solder voltage regulator

This first component that we will solder on the qbcan board is the voltage regulator. This one is one of the difficult parts to solder into the board given how small the component is, so be patient and don't rush it. Familiarise yourself with surface mount soldering (SMD) before attempting to solder this component (check the suggested tutorials) and use a fine pair of pliers (or tweezers better) to place the component in the correct position. The final result is shown in Fig. 9.

1.5 Solder the battery connector

Next on the board we will solder the battery connector. This is also a tricky component to solder as the hole in the board is larger than the metal conductor of the battery connector. Use a third hand (or ask somebody to help you) so that you are able to get everything in position and solder (be careful not to burn your assistant with the soldering iron!). It is also important that you solder the red wire to the + terminal and the black wire to the - terminal (are marked in the board). After soldering you will have something

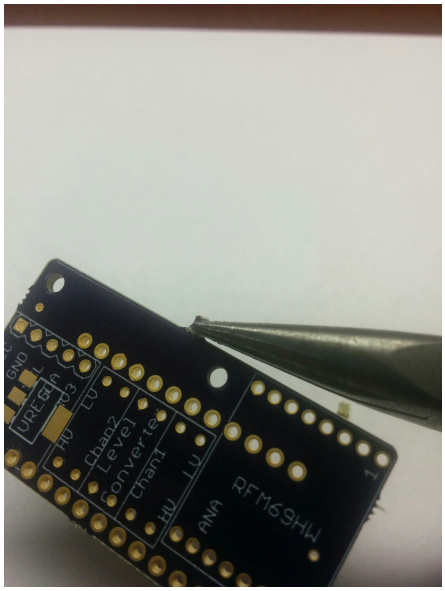


Figure 3: Removing manufacturing leftovers from the board.

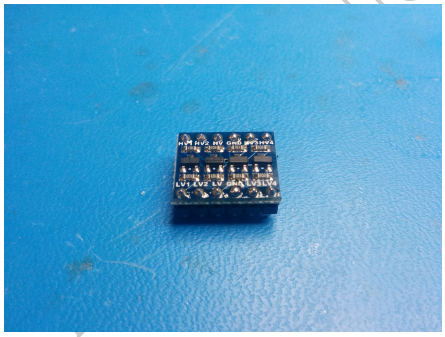


Figure 4: Logic Level Converter with the headers soldered.

similar to Fig. 10. . Note that the wire can protrude significantly (as in Fig 10), if that is the case, cut the protruding edges (see Fig.).

After finishing the soldering clean the connection, plug the 9V battery and with a multimeter check that the voltages that come out of the voltage regulator are as expected (check the Schematic and the Board Layout).

1.6 Solder the Logic Level Converter and the BMP180 into the board

When the the connections of the battery and the voltage regulator have been verified it is time to solder the Logic Level Converter and the BMP180 into the board. This is an easy step but make sure that you solder this two components in the correct orientation. For the Logic Level Converter check that the High Voltage (HV) and Low Voltage (LV) sides are oriented correctly (look at the board markings). The BMP180 needs to fit inside the board (see the board marking for the outline of where the BMP180 should go). The final result after this step is shown in Fig. 12.

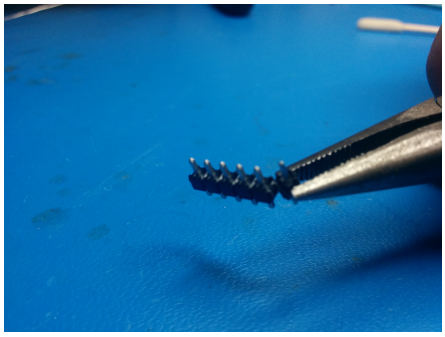


Figure 5: Break one of the pins in the header so it has only 5 pins.

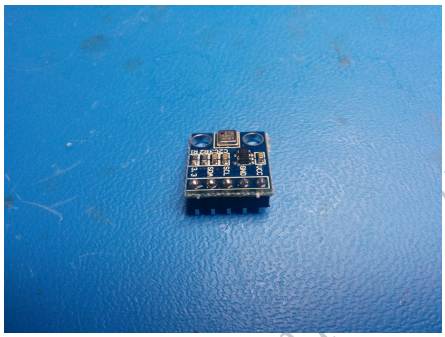


Figure 6: Pressure and temperature BMP180 sensors with headers.

1.7 Solder the RFM69 headers into the board.

Next we will solder the RFM69 headers into the board (**don't solder the RFM69 to the headers yet!**). The headers in this case have 2 mm pitch (are different than the ones used before).

Again, the orientation of the headers does not matter - just make sure that the spacer is located on the correct side of the board (the one with the RFM69 markings). Also it is very important that you **do not solder the antenna pin!**

With the pliers remove the antenna pin from the header. In the board the antenna location is marked with `ANA`. After this step you board should look as in Fig. 13.

1.8 Solder the Arduino to the board.

The next step is to solder the Arduino the board. This should not be very difficult. Refer to the marking on the board to know how the Arduino is oriented in the board. Figure 14 shoes how the board looks like after completing this step.

1.9 Solder the antenna to the RFM69

The RFM69 is a 433 MHz transceiver and we will use a wire monopole antenna. This is the simplest type of antenna consisting of a piece of wire that is a quarter of the wavelength λ. For a frequency f of 433 MHz the length of the antenna l can be easily computed using Eq. 1. The speed of light is represented by the term $c = 299792458\,m/s$. For 433 MHz $l \approx 17.3\,\text{cm}$.

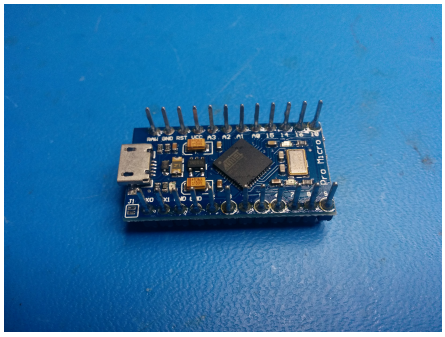


Figure 7: Arduino Pro Micro with headers. Note how the long pins are located on the top of the board..

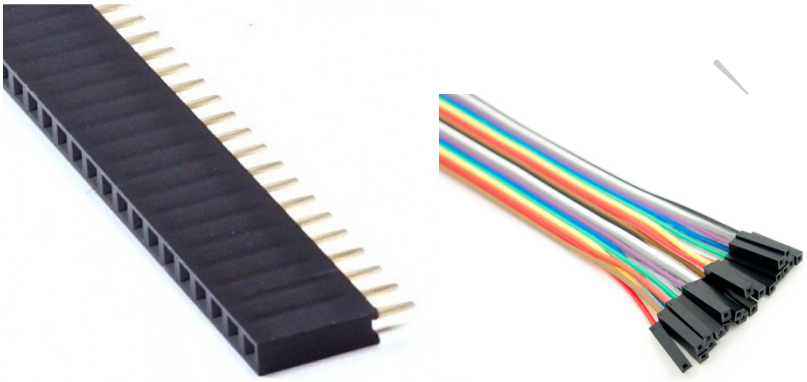


Figure 8: Left - Female headers. Left - Female jumper wires.

$$l = \frac{\lambda}{4} = \frac{c}{4f} \tag{1}$$

Cut a wire to the specified length and the solder it to the RFM69 transceiver. The antenna wire should stick out from the top side of the RFM69. The result should look like Fig. 15.

It is important for the antenna to be as straight as possible. If it is curved or bend the performance and the range will decrease.

1.10 Solder the RFM69 into the board

Finally you can solder the RFM69 into the board (using the headers previously soldered). Once done you will have something like Fig. 16 and the electronics of your qbcan will be completely assembled!

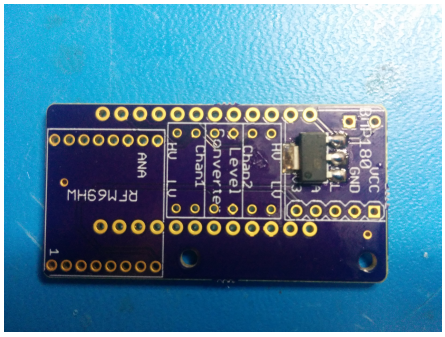


Figure 9: Voltage regulator soldered onto the board.

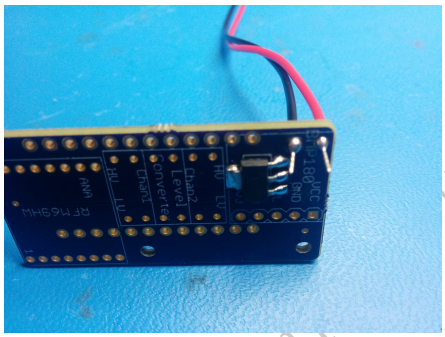


Figure 10: Battery connector just soldered.

2 Software Installation

The Arduino Pro Micro microcontroller is the core of the qbcan but it will need to software in order to execute its mission. In this section you will learn to set-up a development PC where you can develop and upload code to your qbcan. The following steps will guide through the whole process.

2.1 Install the Arduino IDE

The core of QBcan is an Arduino Pro Micro. Arduinos are a family of open-source microcontroller boards. In order to program the Arduino we need to install its Integrated Development Environment (IDE).

At the Arduino software page you can find the latest IDE. Download the version for your platform and install it. Arduinos are programmed in C and using this IDE you will be able to compile the code and upload it to the Arduino. The Arduinos are very popular a lots of tutorials and forums exist online. If you don't have any experience in programming C or Arduinos we would recommend you to watch some of the Jeremy Blum Arduino tutorials.

2.2 Install the Pro Micro drivers

The Arduino IDE works out-of the box with most of the Arduino boards but not with the Pro micro. To make it work additional drivers need to be installed.

Follow this guide to install the drivers for Windows, or this guide to install the drivers for Mac/Linux. As the Arduino Pro Micro is very similar to the Arduino Fio in some instances selecting the Arduino Fio board

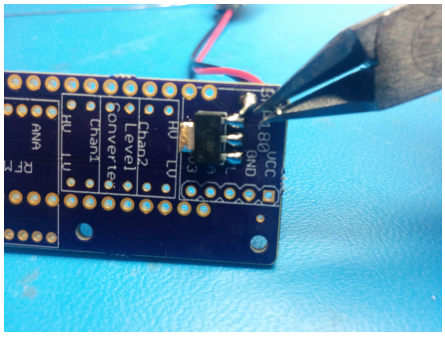


Figure 11: Cut any wire remaining that is protruding.

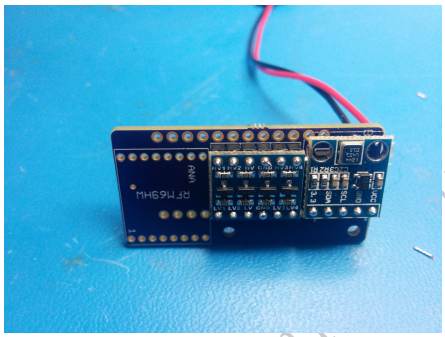


Figure 12: Logic Level Converter and BMP180 solder into the board.

(in `Tools > Board`) will work. So if you are having trouble installing the drivers try this option instead (it might just work fine!).

2.3 Check your installation

Now you should be ready to start programming the Pro Micro and it is now a good time to check that everything is installed correctly.

To test the installation you don't need to have qbcan assembled. Just pick the Pro Micro and plug it to your computer using a micro USB device.

Using the Arduino IDE you can upload this sketch to check that the whole set-up is working (you will see two LEDs blinking). If that works then you are ready to start using qbcan as your CanSat bus

2.4 Install the qbcan Arduino Library.

qbcan comes with an Arduino Library that helps you interface with the transceiver and the pressure and temperature sensor. To install the library just go into the Arduino IDE and click to the `Sketch` menu and then `Include Library > Manage Libraries`. At the top of the drop down list, select the option `Add .ZIP Library`. Then browse to `QBcan_Library.zip`.

Then you will need to close and start again the Arduino IDE for the library to be loaded. To check that the library has been correctly installed you should be able to see the qbcan examples in `File > Examples`.

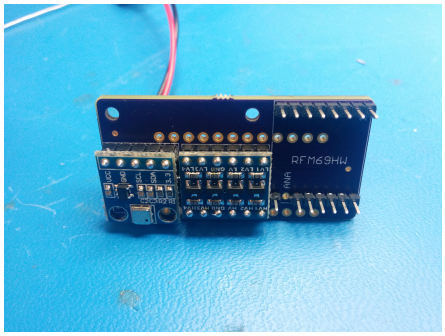


Figure 13: Headers of the RFM69 soldered into the board, note how the antenna pin has not been included.

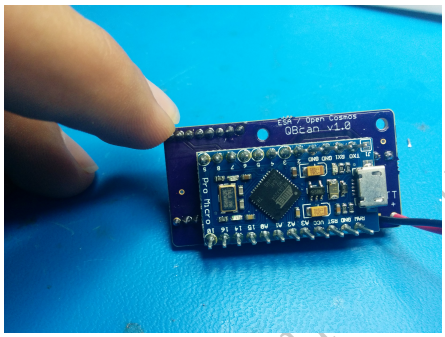


Figure 14: Arduino soldered into the qbcan board.

2.5 Test qbcan using the qbcan Library examples.

The qbcan Library comes with examples you can use to check that the whole system works. To get the examples in the Arduino IDE click on `File > Examples`. You have a `CanSat` examples that measures temperature and pressure and transmits it. The `Ground _Station` example receives this data (open the serial monitor in `Tools > Serial` Monitor to view the received data.) . If you run them on two different qbcan you should be able to send and receive data and test the

To test this examples you will need a completely assembled qbcan. Also we encourage you to use this examples as the starting point for your projects.

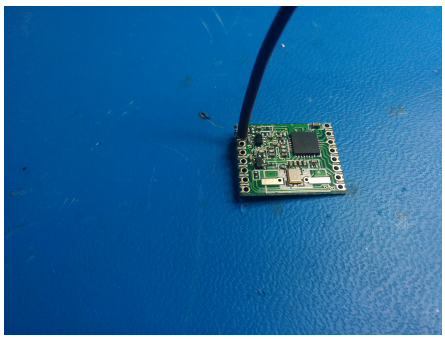


Figure 15: RFM69 with 17.3 cm wire antenna.

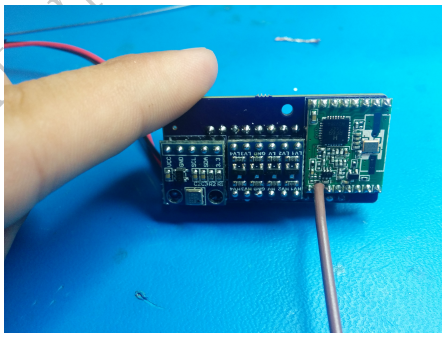


Figure 16: RFM69 soldered into the qbcan.

Part III
Library

An very important part of qbcan is its library. This library provide an easy-to-use interface to the transceiver and the pressure and temperature sensor. In Sec. 2 a step-by-step guide on how to install the qbcan library can be found.

1 Library basics

To use the library at the top of your sketch you will need to include the qbcan library and the `SPI.h` and `Wire.h` libraries (so that we have access to the SPI and I2C buses).

```
//Include the required libraries
#include <QBcan.h> //Core qbcan library
#include <Wire.h> //I2C bus library
#include <SPI.h> //SPI bus library
```

The `SPI.h` and `Wire.h` libraries are part of the Arduino IDE and thus no installation is required. The library `#include` have to be placed at the top of the sketch.

2 Using the BMP180 pressure and temperature sensor

The qbcan library provides a class to interface with the BMP180 pressure. To create the sensor object use the following snippet.

```
//Create the pressure and temperature sensor object
BMP180 bmp;
```

Use this piece of code just after the library `#include`. This will create the sensor object.

On the `setup()` function you will need to initialise the sensor. You can do this as follows.

```
void setup()
{

  //Initialize serial connection for debugging
  Serial.begin(9600);

  // Initialize pressure sensor.
  if (bmp.begin())
    Serial.println("BMP180 init success");
  else
  {
      //In case of error let user know of the problem
    Serial.println("BMP180 init fail (disconnected?)\n\n");
      while(1); // Pause forever.
  }

}
```

After initialising the sensor it is ready to measure. To measure temperature and pressure just do the following.

```
//Declare temperature and pressure variables
double T,P;

// Get a new temperature and pressure reading
bmp.GetData(T,P)
```

The temperature is measured in degrees Celsius and the pressure in mbar.

After you take the measurement you can send the data to the serial port to display it in your PC serial monitor as follows.

```
//Display data
Serial.print("Absolute pressure: ");
Serial.print(P,2);
Serial.println(" mb.");
Serial.print("Temperature: ");
Serial.print(T,2);
Serial.println(" deg C.");
```

3 Using the RFM69 transceiver

The library also provides a class to interface with the transceiver. You can create the object as follows.

```
//Radio object
RFM69 radio;
```

Before initialising the RFM69 object on the `setup()` function it is useful to define some of the transceiver parameters[3].

```
//Radio Parameters
#define NODEID          2    //unique for each node on same network
#define NETWORKID       100  //the same on all nodes that talk to each other
#define GATEWAYID       1    //Receiving node

#define ENCRYPTKEY      "sampleEncryptKey" //The same on all nodes!
```

The `NETWORKID` can be set from 0 to 255 and that changes the physical channel (i.e. the frequency). Although in this guide the `NETWORKID` is considered a parameter it can be changed during runtime (you just need to re-initialise the library).

The NODEID is just the node in a particular challenge and that can be set from 1 to 255. Therefore we can have up to 255 qbcan operating in the same frequency, although they can not transmit at the same time. Messages are generally send to a specific node and that is the `GATEWAYID`. So if you are configuring a transmitter/receiver make sure that you are transmitting to the correct node. Also a receiver can sniff

[3]Some extra parameters will be included in the final version (as bitrate). Currently other configuration parameters are buried inside the library and so they are not so apparent to the user (although they can be easily changed in the library). There is a bit of work to do to decide which are the parameters that users will use the most. The RFM69HW has quite a few registers.

all the packets in a network, thus receiving all traffic in that network independently if the messages where addressed to that particular node.

To configure the radio on the setup you can just use the following code.

```
void setup() {

  //Initialize serial connection for debugging
  Serial.begin(9600);

  //Initialize radio
  radio.initialize(FREQUENCY,NODEID,NETWORKID);
  radio.setHighPower(); //Use the high power capabilities of the RFM69HW
  radio.encrypt(ENCRYPTKEY);
  Serial.println("Transmitting at 433 Mhz");

}
```

To send data through the network.

```
//Send Data
char payload[50];
sprintf(payload,"T: %d C, P: %d mb.",(int)T,(int)P);    Serial.print(payload);
radio.send(GATEWAYID, payload, 50);
Serial.println("Send complete");
```

On the `radio.send()` the last argument is the length of the message to transmit. In the example above the the message `payload` is 50 bytes long and we are sending it in full although the actual message is not that long (so it could be optimised).

To receive data from a network.

```
if (radio.receiveDone())
{
  for (byte i = 0; i < radio.DATALEN; i++)
  Serial.print((char)radio.DATA[i]);
}
```

If you want to sniff al the packets of the network use:

```
radio.promiscuous(true);
```

There are also some other functions that you may find useful when receiving messages:

- `radio.SENDERID` - returns the sender node id.
- `radio.TARGETID` - returns the message target node id (in case you are sniffing all the packets in the network).
- `radio.RSSI` - returns the received signal strength (RSSI).

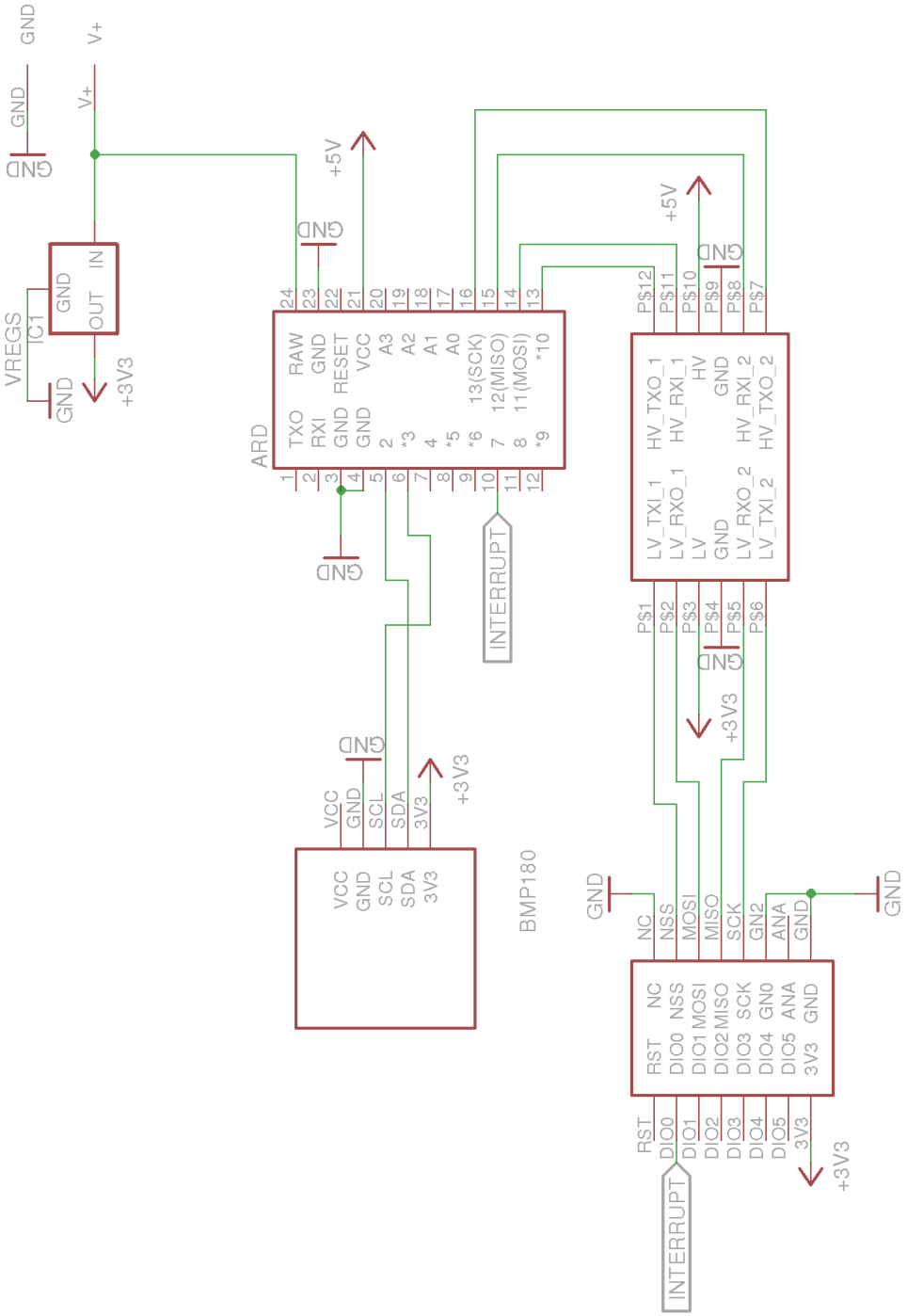
4 Examples

The provided library contains examples that can help you get started. The examples can be found in the Arduino IDE by clicking on `File > Examples > qbcan_Library`.

There is one example, labelled as `CanSat` that transmits data and another one labelled as `GroundStation` that will receive it.

Appendices

Schematic



Board Layout