



The

CAN  **sat**

Book

2013 – Edition

Content

About the CanSat book.....	4
Introduction to CanSat	5
What is a CanSat?.....	5
The CanSat-kit	5
The Primary Mission	6
Getting started	6
Installing software	6
Launch the Arduino application	7
Testing the Arduino Uno board.....	9
Assembling the components.....	10
Constructing the CanSat-shield	10
Testing the CanSat kit.....	13
Using the sensors	16
Intro	16
Analogue to Digital	16
Sensors	17
Calibrating the sensors.....	21
Altitude Calculations	22
Using the radio	24
Introduction.....	24
Transceiver	24
Setting up transceiver hardware	25
Installing drivers	25
Programming the transceivers with Arduino	25
RF-Magic.....	27
Preparing CanSat Shield Radio	29
Testing the radio.....	29
Tera Term VT	29
Parachute design	31
Required Descent Parameters.....	31
Parachute production.....	31

Descent Physics	32
Semi-spherical Parachute Design	33
Cross Parachute Design	34
Parapent	34
Flat parachute design	35
CanSat Design	36
Introduction.....	36
Competition requirements.....	36
The Bracket.....	37
Antenna design.....	38
Appendix.....	39
Frequencies	39
Component list	40
Simulation program "Launch"	42

About the CanSat book

This book is written by Thomas Gansmoe, Stian Vik Mathisen, and Jøran Grande from NAROM together with Jens F. Dalsgaard Nielsen from Aalborg University and Nils Kristian Rossing from the Norwegian University of Science and Technology. The CanSat shield used in this book is developed by Jens F. Dalsgaard Nielsen and Simon Jensen from Aalborg University College.

The CanSat book is built up so that you can start from scratch and get a feeling of mastering the kit as you read through the book. In the beginning we will describe how you can get your Arduino board and shield up and running. Then we will turn over for the missions. In this book we have described a primary mission and how you will get data from it. In some competitions you have a standard primary mission that is equal to all. Secondary mission will be an optional mission that your team is creating.

Introduction to CanSat

What is a CanSat?

A CanSat is a simulation of a real satellite, integrated within the volume and shape of a soft drink can. The challenge for the students is to fit all the major subsystems found in a satellite, such as power, sensors and a communication system, into this minimal volume. The CanSat is then launched to an altitude of a few hundred meters by a rocket or dropped from a platform or captive balloon and its mission begins: to carry out a scientific experiment and achieve a safe landing.

CanSats offer a unique opportunity for students to have a first practical experience of a real space project. They are responsible for all aspects: designing the CanSat, selecting its mission, integrating the components, testing, preparing for launch and then analysing the data.

The CanSat-kit

The CanSat-kit is build up by an Arduino Uno board and a sensor shield board. Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments. Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators.

Here are some examples on Arduino projects:

- Make an automatic night light which switches on when its dark
- Intrusion alarm
- Thermostat
- Line Follower Robot
- Fountain and/or lights that respond "happy to see you" via proximity and/or motion sensors.
- Ham radio Morse code keyed/propagation beacon.
- Graphical calculator that graphs serial inputs on a graphical LCD.
- Wi-Fi controlled RC-Car

Many more ideas can be found at <http://arduino.cc/playground/Projects/Ideas>

The sensor shield card has been developed at the University of Aalborg by Professor Jens Dalsgaard Nielsen and Simon Jensen. The shield has been designed to include:

- Communication radio (APC220) with antenna
- Pressure sensor (MPX4115)
- Temperature sensor (LM35DZ)
- Temperature sensor (NTCLE100E3103JB)
- Three axis accelerometer (MMA7361L)
- SD storage card (OpenLog)

The sensor shield is designed to fit on top of the Arduino Uno R3 board.

The Primary Mission

Getting started

The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analogue inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

To get started you have your ARDUINO UNO board in front of you, a computer with administrative rights and a standard USB cable (A plug to B plug): the kind you would connect to a USB printer.

The open-source Arduino environment makes it easy to write code and upload it to the I/O board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing, AVR-gcc, and other open source software.

Connecting the board

The Arduino Uno automatically draws power from either the USB connection to the computer or an external power supply. Connect the Arduino board to your computer using the USB cable. The green power LED (labelled PWR) should go on.

Installing software

You can download the latest release of the Arduino software by going to <http://arduino.cc> and click the "Download" link in the top menu. Here you can choose a download depending on what operating system you use.

- Click on the downloaded zip-file
- Extract the contents to a folder on your computer, for example "*C:/Program files/Arduino*".
- The content will then be put in a subfolder of this location

Install drivers

Installing drivers for the Arduino Uno with Windows 7, Vista or XP

- Plug in your board and wait for Windows to begin its driver installation process. On some computers the drivers will install automatically. If not, follow the steps below.
- Click on the Start Menu, and open the *Control Panel*.
- While in the *Control Panel*, navigate to System and Security. Next, open the *Device Manager* which you will find under *System*.
- Look under *Ports (COM & LPT)*. You should see an open port named "*Arduino UNO (COMxx)*". In some cases you won't find "*Arduino Uno*", instead you will find "Unknown device" at the top.
- Right click on the "*Arduino UNO (COMxx)*" port and choose the "*Update Driver Software*" option.
- Next, choose the "*Browse my computer for Driver software*" option.

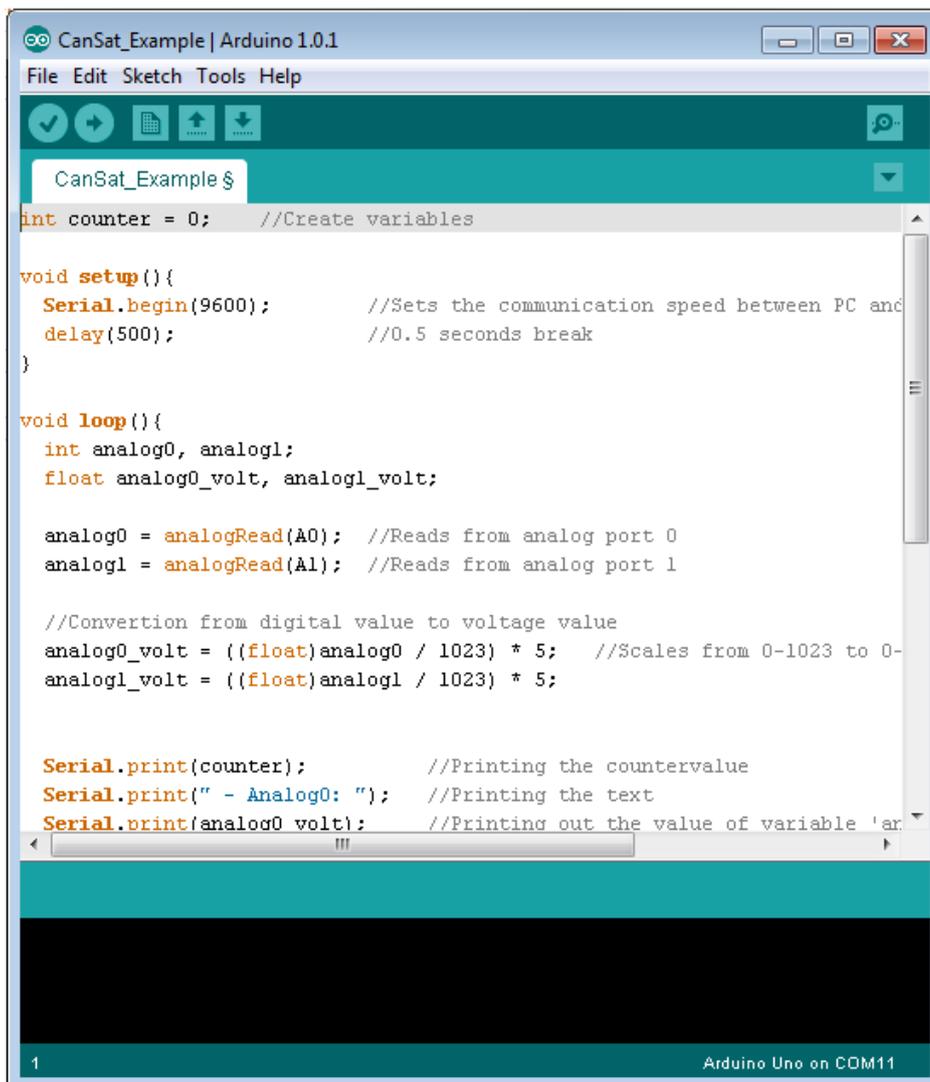
- Finally, navigate to and select the Arduino Uno's driver file, named "ArduinoUNO.inf", located in the "Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" subdirectory).
- Windows will finish up the driver installation from here.

Launch the Arduino application

Double-click the Arduino application in the folder where you extracted it. To find this application easier you can also make a shortcut to the desktop.

Open example code

Open the example code by: File → Open → "CanSat DVD" → ".\code\ArduinoTest\ArduinoTest.ino".



```

CanSat_Example | Arduino 1.0.1
File Edit Sketch Tools Help
CanSat_Example $
int counter = 0; //Create variables

void setup(){
  Serial.begin(9600); //Sets the communication speed between PC and
  delay(500); //0.5 seconds break
}

void loop(){
  int analog0, analog1;
  float analog0_volt, analog1_volt;

  analog0 = analogRead(A0); //Reads from analog port 0
  analog1 = analogRead(A1); //Reads from analog port 1

  //Conversion from digital value to voltage value
  analog0_volt = ((float)analog0 / 1023) * 5; //Scales from 0-1023 to 0-5
  analog1_volt = ((float)analog1 / 1023) * 5;

  Serial.print(counter); //Printing the countervalue
  Serial.print(" - Analog0: "); //Printing the text
  Serial.print(analog0_volt); //Printing out the value of variable 'analog0_volt'
}
1 Arduino Uno on COM11
  
```

Figure 1: Arduino 1.0 with test code

Select your board

You'll need to select the entry in the *Tools* → *Board* menu (Figure 2) that corresponds to your Arduino. Otherwise you won't be able to communicate correctly with the Arduino Uno board.

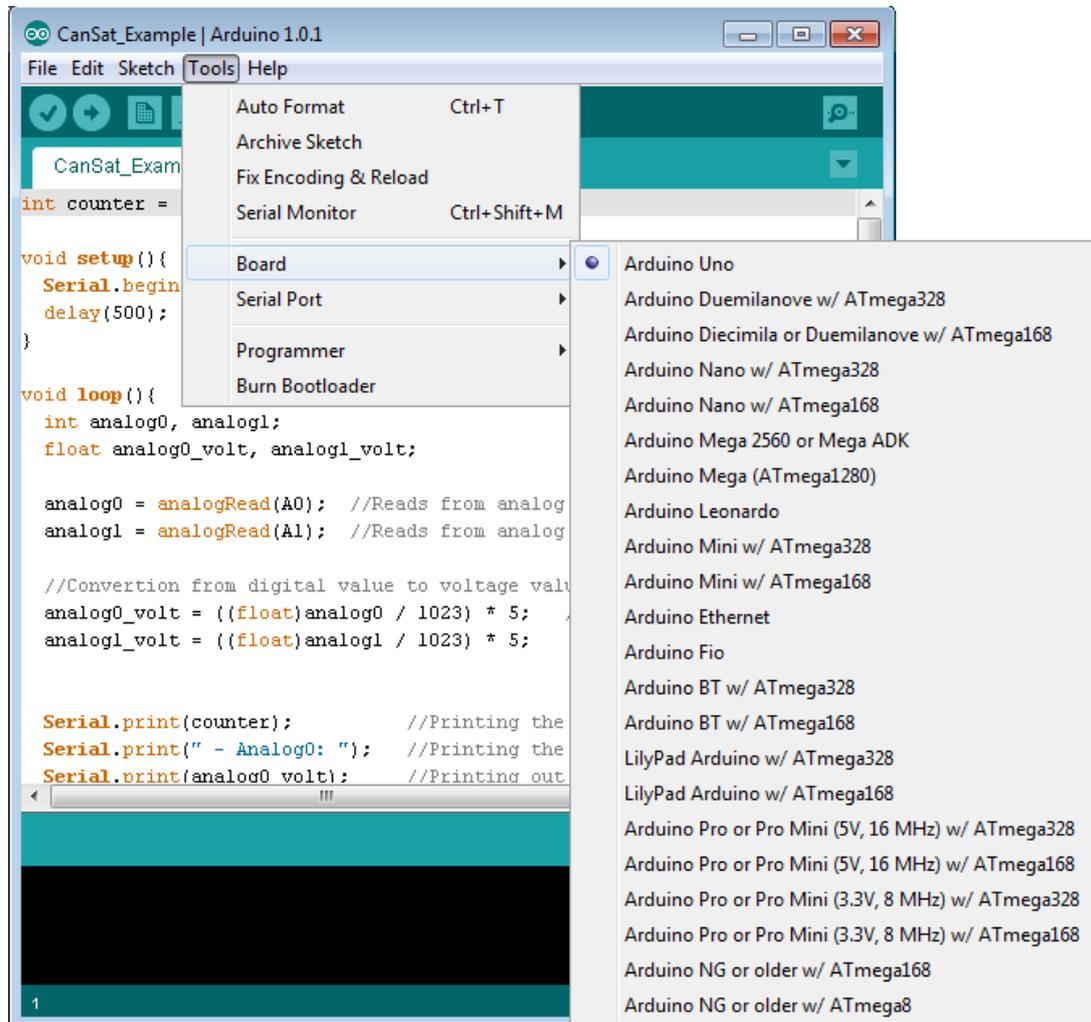


Figure 2: Selecting the Board

Select your serial port

Select the serial device of the Arduino board from the **Tools** → **Serial Port** menu. This is likely to be **COM3** or higher (**COM1** and **COM2** are usually reserved for hardware serial ports). To find out, you can disconnect your Arduino board and re-open the menu; the entry that disappears should be the Arduino board. Reconnect the board and select that serial port.

Uploading the program

Now, simply click the "Upload" button (Figure 3) in the compiler. Wait a few seconds - you should see the RX and TX LEDs on the board flashing. If the upload is successful, the message "Done uploading." will appear in the status bar.

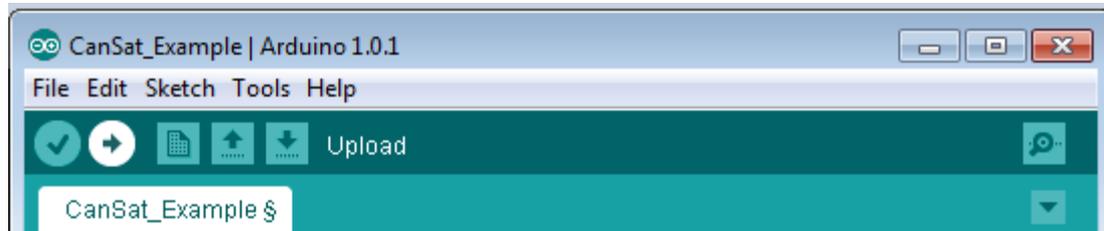


Figure 3: Uploading a code

Read data

Open the Serial Monitor to look at the data which is received from the Arduino Uno board. The Serial Monitor is opened by clicking the icon to the right in the toolbar (Figure 4). The Serial Monitor will open a new window which will give you a serial text stream of the data printed from the Arduino board.

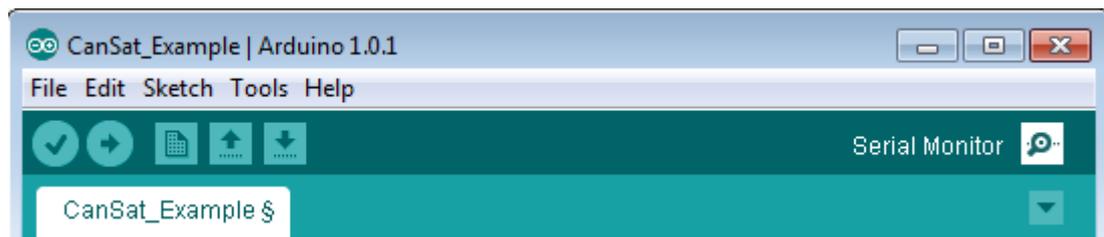


Figure 4: Opening the Serial Monitor

Testing the Arduino Uno board

To verify that you are receiving correct data you can test it by setting each channel to ground and power and read the output in the Serial Monitor. To do this you need a wire to connect between the analogue connectors and power connectors on the Arduino Uno board.

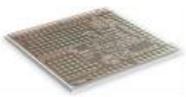
- Connect one end of the wire to **A0** port
- Connect the other end to **GND** port
- Analog0 in the Serial Monitor should now read 0.0 volts
- Remove the wire from **GND** and connect it to **5V**
- Analog0 should now read approximately 5 volts
- Remove the wire from **5V** and connect it to **3.3V**
- Analog0 should now read approximately 3.3 volts
- Repeat the same procedure with the **A1** port

Assembling the components

Constructing the CanSat-shield

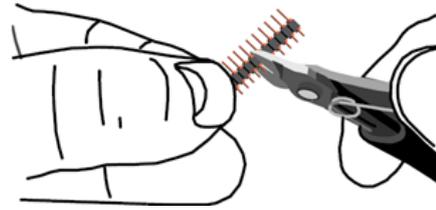
This section will show you step by step how to assemble the components for the CanSat shield.

Part list

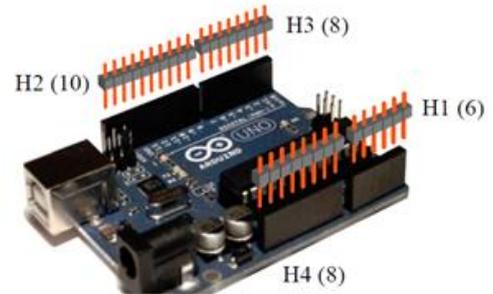
Arduino Uno R3	
CanSat Shield (Circuit board)	
OpenLog Data logger with SD-card	
APC220 Radio transmitter/receiver	
Temperature sensors LM35DZ and NTC (NTCLE100E3103JB0)	
MPX4115A Pressure sensor	
MMA7361L Accelerometer 3-axis	
Resistor 1x 75 Ohm 1x 10 kOhm	
Male pin connector header 4 x 16-pin	
Female pin connector header 7-pin and 2-pin	

Assembling guide

1. Cut the male pin connector header in to the following lengths:
 - 6 pins (H1)
 - 10 pins (H2)
 - 8 pins (H3)
 - 8 pins (H4)

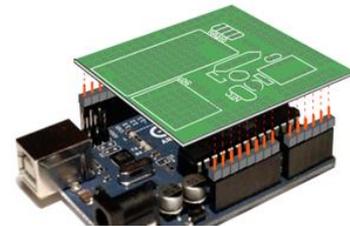


2. Insert the connector headers into the Arduino board with the short end up.

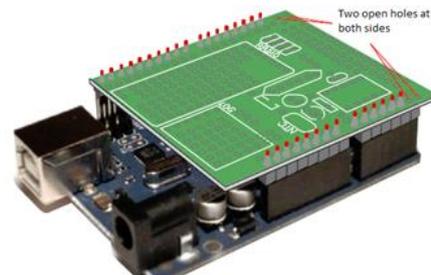


3. Mount the shield board on top of the Arduino Uno.

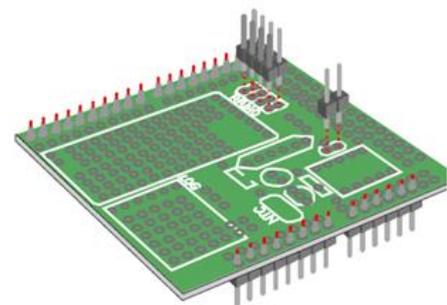
Note: The board should fit only one way.



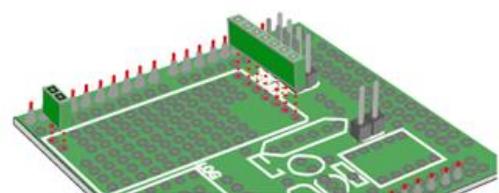
4. Solder all the pins on the top of the circuit board and then remove it from the Arduino Uno. Make sure not to heat the pins to long while soldering. To long exposure to heat might damage the Arduino board.



5. Cut out two lengths of 4-pin connector header and one length of 2-pin connector header. Place them on the top of the shield board and solder them on the bottom(soldering side) as shown in the illustration. (Placed at the J1, J2 and J3 position)

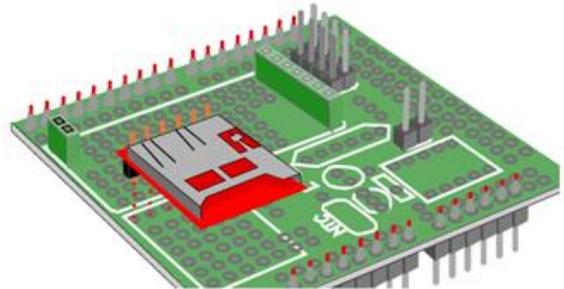


6. The green female connector header allows us to easily connect and disconnect the radio for programming. If you are unfamiliar with this kit, it is highly recommended to use the green connector. This may if necessary be removed to make it more robust.

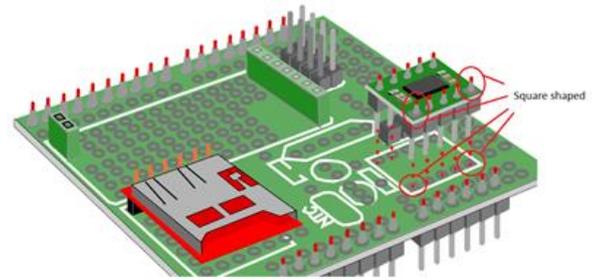


Place the connectors on top of the board and solder them at the soldering side.

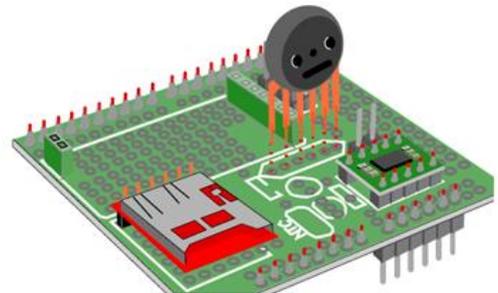
7. Solder the data logger onto the shield board using a 6-pin connector header. Use the short end of the connector downwards through the shield board. Make sure you place the logger the correct way (See illustration). It is recommended to use some hot glue in between the board and the logger to support the logger. If not supported it may easily brake off or damage the soldering points.



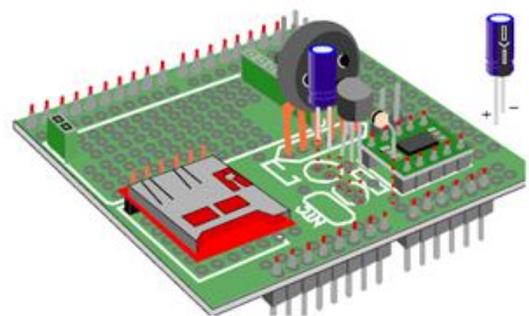
8. Use two lengths of 5 pin male connector header and solder the accelerometer board onto the shield board at the U1- position. Make sure to put the sensor the correct way. The black IC-chip on the accelerometer board should be on the top (pointing upwards). Also note that two of the solder pads both on the shield board and the sensor have a square shape instead of a circle shape. These squares should be aligned.



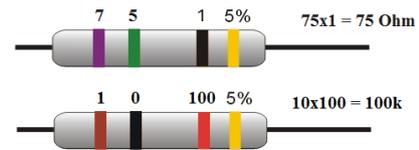
9. Put the pressure sensor onto the shield board and solder it on the bottom side. Make sure to put the sensor the right way (As in the illustration).



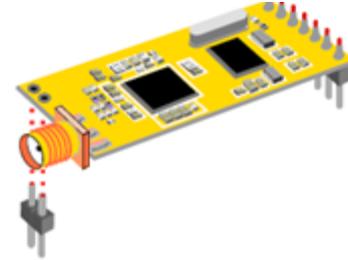
10. Continue by soldering the 75 Ohms resistor to the R2 position, the 1uP capacitor to C1 position and the temperature IC sensor (LM35DZ) to T1 position. NB! The label showing the orientation of the temperature sensor T1 on the board is wrong. It should be turned 180 degrees. The capacitors orientation is labelled with a plus sign (+) on the board. The longest pin on the capacitor is the positive one. The negative pin is also labeled on the side of the capacitor.



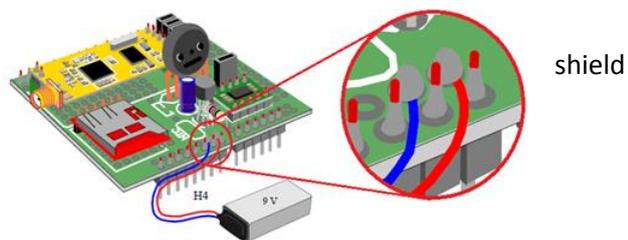
Note: There are two different resistors in the kit. Make sure you use the 75 Ohm resistor, not the 10k Ohm resistor.



11. Solder a 2-pin male connector header to one of the radios. These pins will only be used as support to reduce the strain on the radio connector. Mount the radio to the shield board.



12. Solder the battery connector to the board.



The CanSat shield board is now ready to be tested. There are still two components which have not been installed; the NTC temperature sensor and the 10k Ohm resistor. The NTC temperature sensor is optional and can be installed by choice.

Note: If you choose to use the NTC sensor you will also have to install the 10k Ohm resistor (R1) to make the sensor work.

Testing the CanSat kit

After finishing the CanSat Shield board we need to test it to make sure that all of the components are working properly. The following steps will guide you through the test procedure:

1. Install the Shield board on top of Arduino Uno (Make sure the power is disconnected before mounting the Shield board).
2. Power up the CanSat by connecting the battery.

There is a green light on the Arduino Uno labelled "ON". This should always be illuminated when the Arduino Uno is powered.

If you fail to get the power light on, test your battery and make sure there are no short circuits on the Shield board.
3. Disconnect the battery, then connect the USB cable and check the "ON" light again.
4. Open the sketch file "*ShieldTest.ino*" in the Arduino editor (1.0 or later version) and upload the sketch to the CanSat kit. This is a test program designed to give you measurements from all the sensors within the CanSat kit.
5. Start the "Serial Monitor" in the compiler.

You should get some data similar to what you see in the illustration below.

```

Counter: 0 | Time[s]: 0.00 | Temp: 2.10 V | NTC: 2.06 V | Pressure: 2.01 V | Acceleration [x,y,z]: 1.96 V, 1.91 V, 2.07 V,
Counter: 1 | Time[s]: 0.50 | Temp: 1.98 V | NTC: 2.02 V | Pressure: 1.99 V | Acceleration [x,y,z]: 1.95 V, 1.89 V, 2.04 V,
Counter: 2 | Time[s]: 0.99 | Temp: 1.94 V | NTC: 1.98 V | Pressure: 1.96 V | Acceleration [x,y,z]: 1.93 V, 1.87 V, 2.02 V,
Counter: 3 | Time[s]: 1.50 | Temp: 1.91 V | NTC: 1.96 V | Pressure: 1.94 V | Acceleration [x,y,z]: 1.91 V, 1.86 V, 2.00 V,
Counter: 4 | Time[s]: 2.00 | Temp: 1.88 V | NTC: 1.93 V | Pressure: 1.92 V | Acceleration [x,y,z]: 1.90 V, 1.84 V, 2.03 V,
Counter: 5 | Time[s]: 2.50 | Temp: 1.88 V | NTC: 1.93 V | Pressure: 1.91 V | Acceleration [x,y,z]: 1.89 V, 1.84 V, 2.02 V,
Counter: 6 | Time[s]: 3.00 | Temp: 1.87 V | NTC: 1.92 V | Pressure: 1.91 V | Acceleration [x,y,z]: 1.88 V, 1.82 V, 2.01 V,
Counter: 7 | Time[s]: 3.50 | Temp: 1.87 V | NTC: 1.92 V | Pressure: 1.90 V | Acceleration [x,y,z]: 1.88 V, 1.83 V, 2.00 V,
Counter: 8 | Time[s]: 4.00 | Temp: 1.85 V | NTC: 1.91 V | Pressure: 1.90 V | Acceleration [x,y,z]: 1.87 V, 1.82 V, 1.97 V,
Counter: 9 | Time[s]: 4.50 | Temp: 1.87 V | NTC: 1.92 V | Pressure: 1.90 V | Acceleration [x,y,z]: 1.87 V, 1.82 V, 1.96 V,

```

6. Description of the measured data

Counter: This will count the number of measurements done by the CanSat.

Time: This is the time in seconds since the CanSat was turned on.

Temp: Shows data from the first temperature sensor (LM35DZ).

NTC: Shows data from the optional temperature sensor (NTC).

Pressure: Shows data from the pressure sensor (MPX4115A).

Acceleration: Show data from the accelerometer (MMA7361L). This sensor will give you three readings, one for each axis (x, y and Z).

By default the sampling speed will be at 2 Hz (Two lines per second) and all the measured values in volts.

7. The test program can be controlled by sending a character via the serial monitor. Enter a character at the command line and press enter.

The test program will accept the following commands:

“1” – Sets the sampling speed to 1 Hz (1 per second).

“2” – Sets the sampling speed to 2 Hz (2 per second).

“5” – Sets the sampling speed to 5 Hz (5 per second).

“R” – Resets the counter back to zero.

“V” – Changes all the sensor outputs to volts.

“S” – Changes all the sensor outputs to measured values in Celsius, Pascal and G.

8. Test all the sensors by trying to manipulate the sensors.

Do you get response from all the sensors?

9. Test the accelerometer setting. Put a shunt on the J1 port. This will regulate the sensitivity for the accelerometer from 1,5G to 6G. The output should be the same, but with a different resolution.

If you have got data from all the sensors and they seem to work fine the testing of the main functionality is done. The radio and data logger will be tested at a later point in this manual.

Note: The sensors may have some deviations from the actual conditions caused by offset in the sensors. There is a section on how to calibrate the sensor further down in this manual.

Notes:

Using the sensors

Intro

The CanSat kit used in this manual comes with a sensor board. Connected to this board are four sensors; a *pressure sensor*, two *temperature sensors* and a *three axis accelerometer*. These sensors produce a voltage depending on the value of the parameter the sensor measures. It can take on any value in a certain range; such a signal is called an analogue signal, viewed at the top in Figure 5. From the measured voltages, we can calculate the corresponding value, for instance converting voltage to temperature.

This section will show you how to convert the analogue voltage measured at the sensor, to the physical quantity with the correct unit.

Analogue to Digital

All digital components operate with discrete signals. Unlike the analogue signal, a signal that can only take on some discrete values is called a digital signal. In Figure 5 these two different signals are illustrated. The top graph shows an analogue (continuous) signal, and the bottom graph shows a digital (discrete) signal.

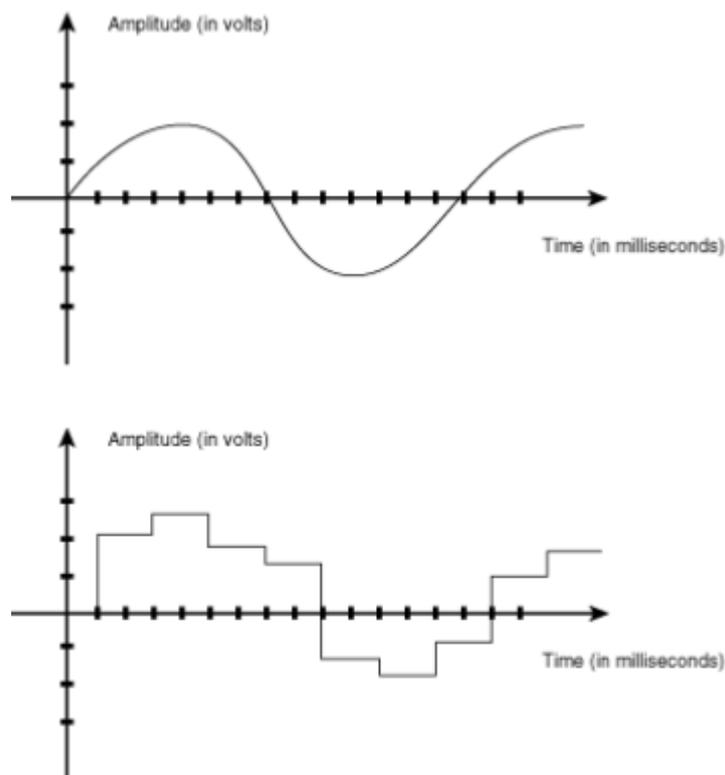


Figure 5: Analogue and digital signal

Desktop or laptop computers, as well as the small computer in the CanSat (called a microcontroller), can only process digital signals. To convert the analogue signal from the sensor into a digital one we

use an Analogue to Digital Converter (ADC), which as the name implies, converts an analogue signal into a digital signal.

The ADC converter is incorporated in the microcontroller and has 8 input channels. It is a 10 bit ADC; it will convert an analogue signal into a digital signal with a 10 bit binary number. One Bit represents one binary digit and can have a value of 0 or 1, which also can be represented by *High* or *Low voltage*, as well as *On* or *Off*. In programming 0 and 1 is also often represented by *True* (1) or *False* (0).

Each digit in the binary number can have 2 values, 0 or 1. A 10 bit binary number can have $2^{10} = 1024$ different values, and can represent an integer ranging from 0 to 1023. The microcontroller can understand digital numbers and use it for computations, which can be programmed into the processor by writing a program code. An example of such a code is the TestShield-code which was used in the previous part of this manual.

Each sensor in the CanSat is sampled by the ADC, which turns each analogue value into a 10 bit number. 0 volts is converted into the binary number 0000000000 = 0 and 5 volts into the binary number 1111111111 = 1023.

By representing the 5 volt input by 1024 levels we have a resolution of $5V/1023 = 4,89mV$. This shows us that with a 10 bit ADC, the smallest voltage change we can measure is 4,89 mV. This is important to notice when we start working with the sensitivity of the sensors.

The sequence of events is thus as follows:

1. The temperature sensor converts the measured temperature into a voltage. This is an analogue signal.
2. The ADC converts the analogue signal into a digital signal, which the processor can handle.
3. Inside the microcontroller the signal is stored as a 10 bit binary number, which can be used for computations.

Sensors

In the “Primary mission” part of this manual we will go through the pressure sensor and both of the temperature sensors. The accelerometer is not a part of the primary mission, and will therefore be described in the “Secondary mission” part of the manual.

Pressure Sensor (MPX4115a)

The pressure sensor used is the MPX4115A from Motorola. It uses a silicon piezoresistive sensor element. Figure 6 presents a cross-sectional view of the sensor.

If a material is called “piezoresistive”, it means that the resistance of the material will change when a mechanical stress is applied. In this case the piezoresistive material is silicon. The changes of

resistance for silicon are far greater than for example steel, making this material very useful to use as a sensor element in a pressure sensor.

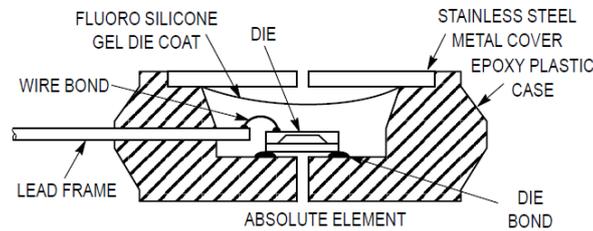


Figure 6: Cross-sectional view of the pressure sensor

Figure 7 shows a more detailed look into the sensor. It shows the dimensions of the sensor and the layout of the connections. To relate the measured voltages to ambient pressure values, the transfer function of the sensor is needed. Such a function describes the mathematical relation between the voltage output of the sensor and the equivalent pressure. This function can be found in the datasheets of the sensor. Figure 8 shows the output voltage versus the ambient air pressure for the sensor as we find it in the datasheet. The output voltage V_{out} is:

$$V_{out} = V_s(0.009P - 0.095)$$

Where P is the airpressure in kPa and V_s is the supply voltage.

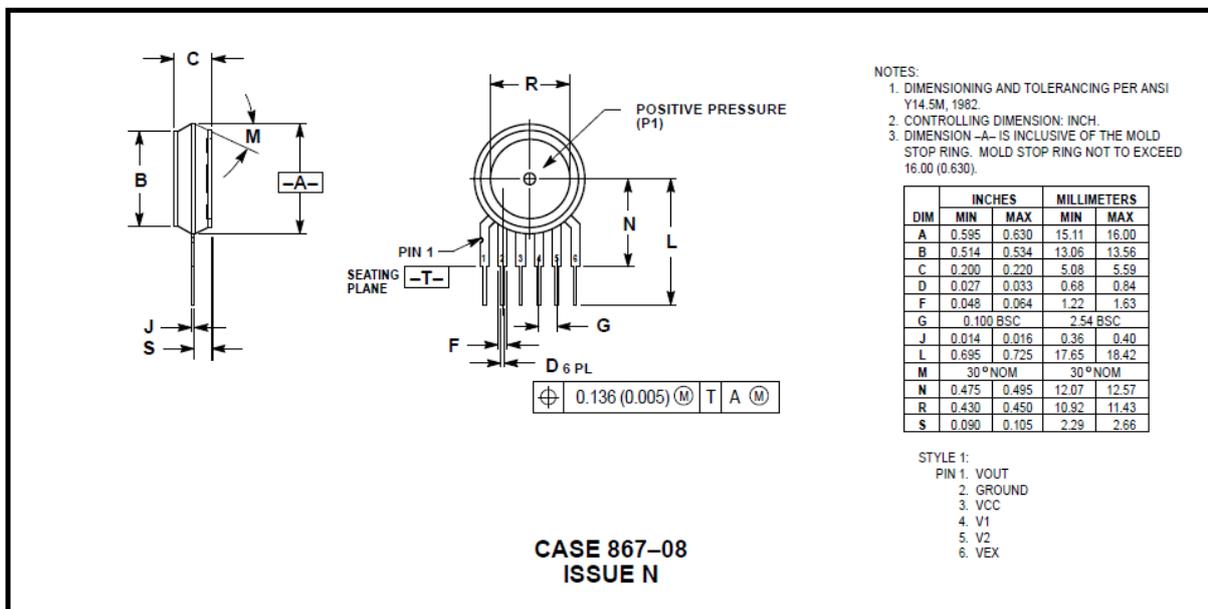


Figure 7: Case style of the pressure sensor

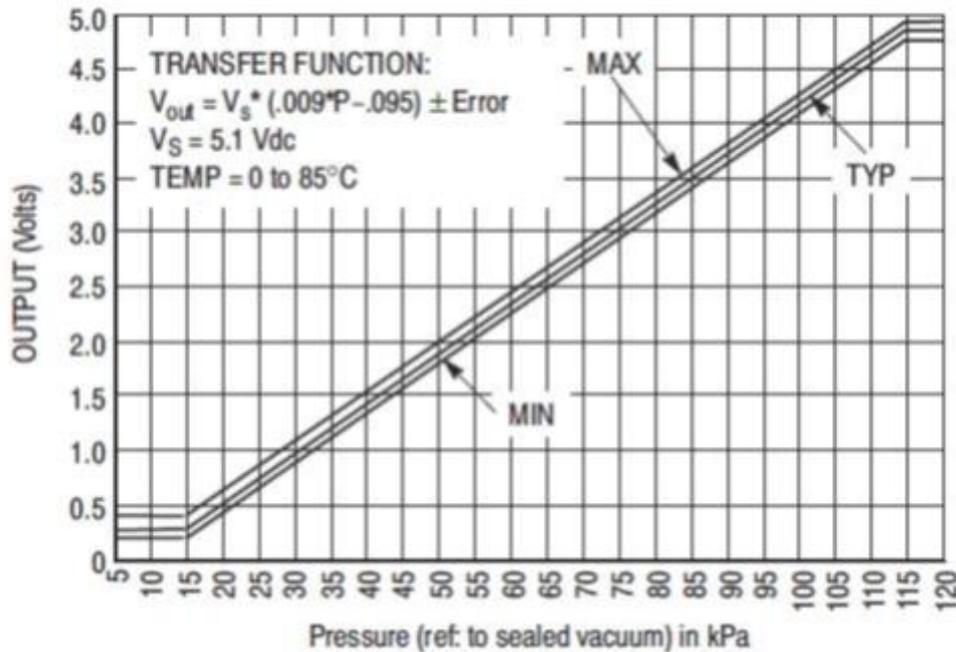


Figure 8: Transfer function of the pressure sensor

Temperature Sensor A (LM35DZ)

The LM35DZ temperature sensor is made for linear measurement in degrees Celsius temperature (or Kelvin). The sensor is made for measuring temperatures between 0°C and 100°C. The output voltage is 250mV at 25°C and the sensor has a sensitivity of 10mV/°C. You can use this information together with the information in the LM35 datasheet, which you will find on your DVD, to make a transfer function of the sensor. You may also plot this on your computer or calculator.

$$\text{Sensor output [V]} = \text{Sensitivity [V/°C]} * \text{Temperature [°C]} + \text{Output at 0°C (Offset)}$$

Temperature sensor B (NTC)

The temperature sensor used in the CanSat is the NTCLE203E3103GB0 manufactured by Vishay/BC components. It is a NTC, or Negative Temperature Coefficient thermistor. The thermal conductivity rises with increasing temperature. Most ceramic materials exhibit such behaviour. Other materials however will have an opposite behaviour, with rising temperature the conductivity decreases. Most NTC thermistors are therefore made out of semi conductive materials, something in between an insulator and a conductor, with some special qualities.

Simply put, when the material is heated, the electrons in the material are energized, so even more electrons are able to move around, thus the material can conduct electricity more easily. When a material can conduct electricity more easily its resistance will decrease. Increased temperature will therefore lead to decreased resistance. This inverse relationship is the reason why this sensor is called a Negative Temperature Coefficient (NTC) resistor.

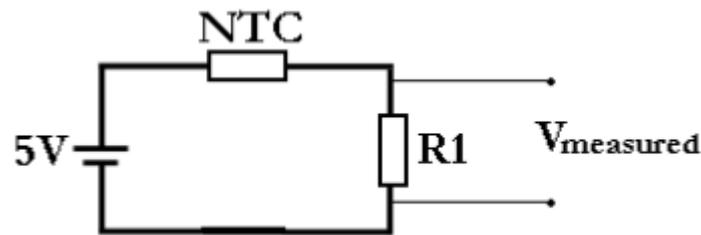


Figure 9: NTC measuring circuit

On the sensor board, the temperature sensor is connected in series with a resistor (R_1) which has a constant resistance of 10 k Ω , as seen in the simplified diagram shown in Figure 9. When resistors are connected in series the current in the circuit will be the same everywhere. The total resistance can be calculated by:

$$R_T = R_1 + R_{NTC}$$

$$I = \frac{U}{R} = \frac{5V}{(R_1 + R_{NTC})}$$

The same current, I , flows through the fixed resistor R_1 giving a voltage $V_{Measured}$ across the resistor. This can be put into the following equation:

$$I_{R1} = \frac{U_{R1}}{R_1} = \frac{V_{Measured}}{R_1}$$

Since the current is the same everywhere in the circuit, we can set up the following equation:

$$I_{R1} = I_{NTC}$$

$$\frac{5V}{(R_1 + R_{NTC})} = \frac{V_{Measured}}{R_1}$$

From this we can get the relation between the measured voltage ($V_{Measured}$) and the resistance of the NTC temperature sensor (R_{NTC}).

$$V_{Measured} = \frac{5V \cdot R_1}{(R_1 + R_{NTC})}$$

$$R_{NTC} = \frac{5V \cdot R_1}{V_{Measured}} - R_1$$

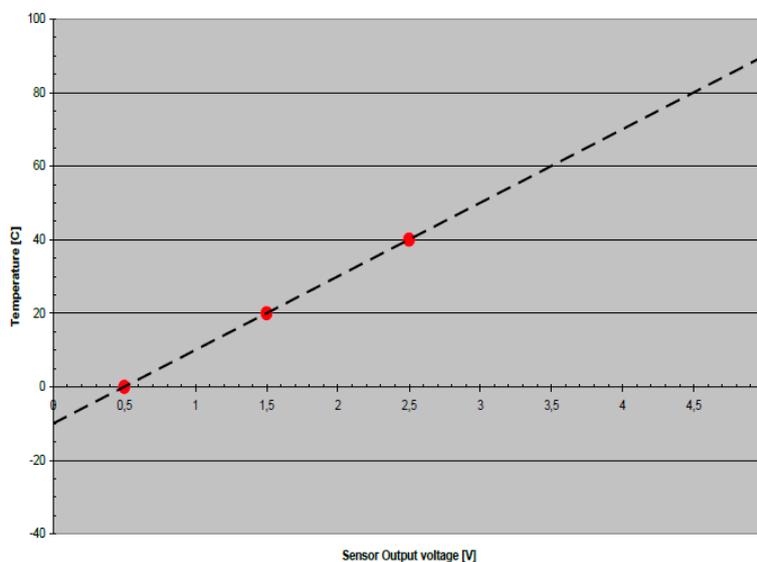
To get a complete transfer function you will also need the relation between the temperature, and the resistance of the sensor (R_{NTC}). You can find this in the sensor datasheet.

Calibrating the sensors

In some cases the transfer function between the temperature and the output voltage of the sensor is unknown. A simple method to determine the transfer function of the sensor, is to measure the output voltage at some (3-4) specific temperatures and plot the result into a graph. For our specific NTC sensor we can assume that the relationship is linear within the range of interest. Linear regression can be used to find the linear equation of best fit.

Linearity means that the relation between voltage and the parameter temperature is directly proportional. This relation can be expressed by the linear formula:

$$\text{Parameter} = A * \text{Voltage} + B$$



Note: This is only an example

Voltage [V]	Temperature [C]
0,5	0
1,5	20
2,5	40

Figure 10: Measured test results

The graph of figure 10 can be used to estimate the values of A and B. In this diagram you can plot the measured voltages on the x-axis, and the parameter (in this case temperature) on the y-axis. In the table next to the diagram you will find some example measurements for this sensor. These measurements are also plotted in the graph. Step two is to draw a straight line through the points. The more points you use the more accurate your result will be. However it will be more difficult to fit a line through each and every data point. Fit the line as closely as possible, and use it to determine the values of A and B from the standard linear formula above.

A: is the slope of the line,

B: is the intersection with the y-axis.

Altitude Calculations

The atmosphere is all around us; it is a thin gaseous layer surrounding our planet. The atmosphere is composed primarily of nitrogen (78%) and oxygen (21%). The remaining 1% consists of water vapour, CO₂ and other trace gasses. The Earth's atmosphere consists of different layers, each one having different properties (temperatures, pressure, composition, etc...).

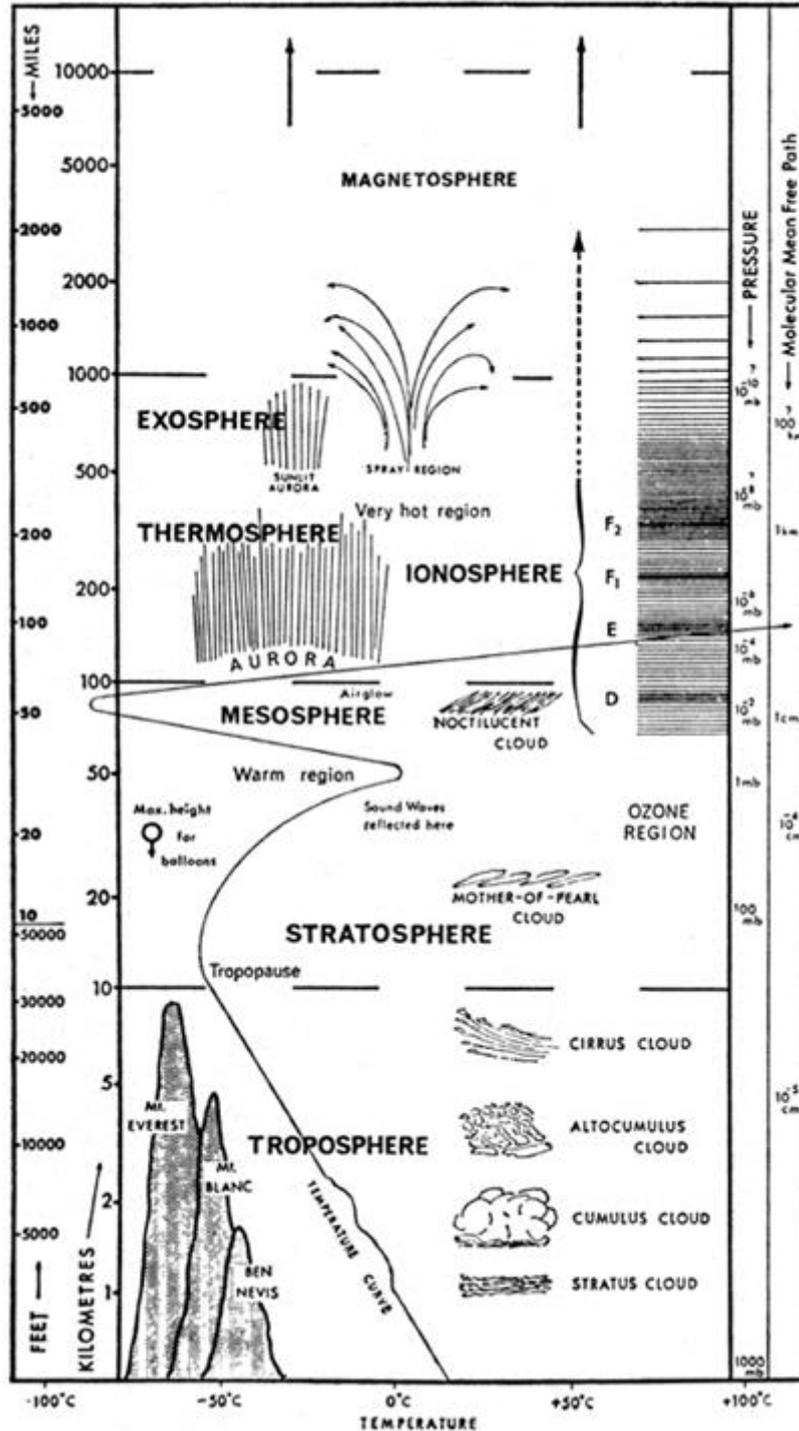


Figure 11: Atmosphere layers

The different layers are represented in Figure 11, along with various human and weather activities seen in these layers.

Unlike our CanSat, most satellites operate in the exosphere. Here the density of the atmosphere is very low. The CanSat however operates in the troposphere, which is the bottom layer of the atmosphere. This layer contains about 80% of the total mass of the atmosphere, and stretches to about 10 kilometres altitude. A great deal of the “weather” we observe on a day-to-day basis (wind and clouds for instance) occurs within this layer.

As seen in Figure 11, the temperature and pressure of the atmosphere varies with altitude.

Although the ambient temperature can rise and fall as you move through the different layers of the atmosphere. Within the troposphere, there is a linear relation between the temperature and altitude. On average, ascending one kilometre from sea level will result in a temperature drop of 6.5 degrees Celsius.

The equation below provides the relation:

- T - Temperature in Kelvin
- T_1 - Starting temperature at h_1 altitude
- h - Altitude in meters
- h_1 - Starting altitude
- α - Temperature gradient: -0.0065 K/m.

The relation between the pressure and the altitude is somewhat more complicated. The pressure is not only dependent on the altitude but also on the temperature. Let’s start with the relation of pressure to temperature:

$$\frac{p}{p_1} = \left(\frac{T}{T_1}\right)^{-\frac{g_0}{\alpha R}}$$

- p - Pressure in Pascal
- p_1 - Start pressure in Pascal
- g_0 - Gravitational acceleration: 9.81 m/s^2
- R - Specific gas constant: $287.06 \text{ J/kg}\cdot\text{K}$

Inserting this formula into the temperature-altitude relation, we achieve the following expression for altitude as a function of temperature and pressure:

$$h = \frac{T_1}{\alpha} \left(\left(\frac{p}{p_1} \right)^{-\frac{\alpha R}{g_0}} - 1 \right) + h_1$$

Using the radio

Introduction

Telemetry is a technology that allows transmission of data from remote measurement devices. It is derived from the Greek words “*tele*”, meaning *remote*, and “*metron*”, meaning *measure*. Telemetry is an essential part of rocketry and satellite technology. Information is transmitted wirelessly using radio waves. On the ground these signals are collected by radio receivers. Large space agencies have networks of these ground stations stretching all over the globe, tracking, monitoring and receiving telemetry from their satellites.

Telemetry data can be divided into two groups: data from internal sources and data from external sources. Rockets and satellites are equipped with countless sensors that measure internal parameters. The measurements they take may relate to temperature, pressure, attitude, power usage, and a wide variety of other measurements. The information from these sensors is called “*housekeeping data*”. This is used to monitor the satellites health, and is necessary for the operation of the system.

Information from the external sources is mostly what interest scientists. This is the data collected from the payload. The payload of a research satellite typically takes the form of sensors or other equipment which measures and generates data about our planet, the space environment, the sun, the stars, or any number of other things depending on the mission. This information is called the “*mission data*” or “*scientific data*”. In your CanSat this would be the information from the sensor board. This data is sent to a ground station to be studied by scientists.

CanSat telemetry operations can be broken down into three distinct components: transmitting data, receiving data, and processing data. The transmitter board inside the CanSat collects information and transmits it as a radio signal. This signal is received by the ground station and sent to a laptop, where it is stored before being processed as experimental data.

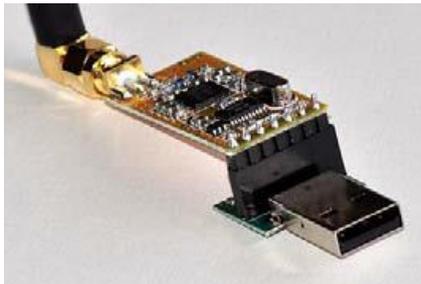
Transceiver

A transceiver is a device comprising both a transmitter and a receiver which are combined and share common circuitry or a single housing. When no circuitry is common between transmit and receive functions, the device is a transmitter-receiver.



In the Arduino CanSat kit we are using APC220 which is a transceiver, with a highly versatile low power radio solution that is easy to setup and integrate into any project requiring a wireless RF link.

It is perfect for robotic application which gives you wireless control. You can connect one of these modules with your microcontroller through TTL interface. And connect your PC with another APC220 module through a TTL/USB converter.



Setting up transceiver hardware

There are two ways of configuring the transceivers. They can be programmed directly from the Arduino board, or you can program them by using the program RF-Magic. RF-Magic has a lot of bugs, and some computers will have problems getting the program to run properly. We will explain both methods, but we

recommend using the Arduino.

Installing drivers

The transceiver at the ground station will be connected to the computer through a USB-TTL converter. This module needs a driver to work properly.

The latest VCP (Virtual Com Port) drivers for the USB-TTL module can be downloaded from Silabs.com (CP210x USB to UART Bridge VCP Drivers).

When you are at silabs.com website, press VCP Driver Kit to get the latest driver and follow the instruction given by the installation wizard.

It's recommended to first install the USB-TTL converter drivers and then plug in the USB converter with the inserted APC module. When the module is inserted, Windows will recognize it and configure the necessary drivers. If Windows does not recognize the device or reports that the device is not functioning properly; remove it from the USB port and reinsert it.

Programming the transceivers with Arduino

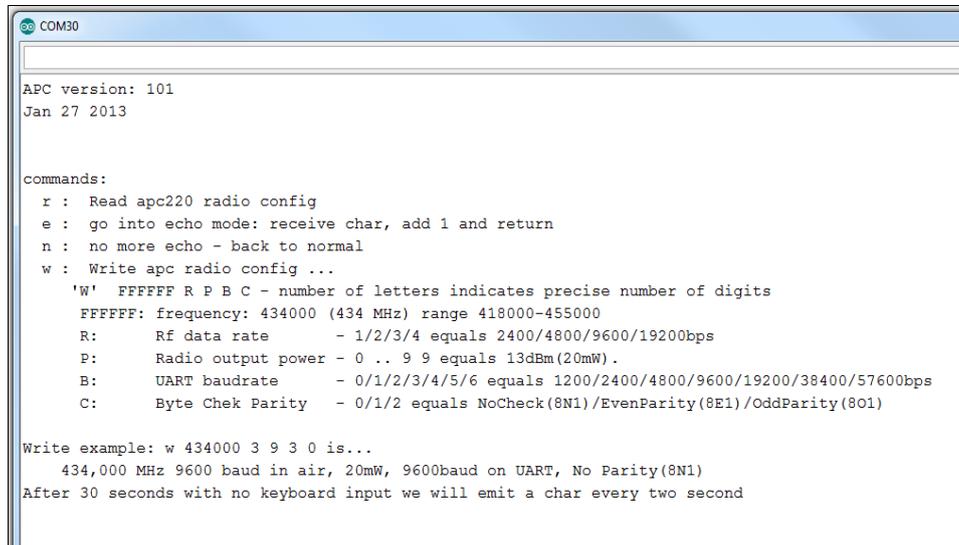
Connect the Arduino board to the computer and upload the program "apc220cfg.ino" which is found on the DVD (or NAROM's web site). Make sure you upload the program before you try to connect the transceiver to the Arduino board. Disconnect the USB cable (and battery) from the Arduino board and connect the transceiver to the Arduino as shown in **Feil! Fant ikke referansebildet..**



Figur 12: Connecting the transceiver to the Arduino UNO board

The transceivers will be connected to the pins labelled GND, 8,9,10,11,12 and 13 on the Arduino board.

Reconnect the Arduino board through the USB cable and open the *Serial Monitor*. In the command line at the top, type in 'm' and hit *enter*. This will bring up the menu shown in Figure 13.



```

COM30
APC version: 101
Jan 27 2013

commands:
  r : Read apc220 radio config
  e : go into echo mode: receive char, add 1 and return
  n : no more echo - back to normal
  w : Write apc radio config ...
      'W' FFFFFFF R P B C - number of letters indicates precise number of digits
      FFFFFFF: frequency: 434000 (434 MHz) range 418000-455000
      R:      Rf data rate      - 1/2/3/4 equals 2400/4800/9600/19200bps
      P:      Radio output power - 0 .. 9 9 equals 13dBm(20mW).
      B:      UART baudrate     - 0/1/2/3/4/5/6 equals 1200/2400/4800/9600/19200/38400/57600bps
      C:      Byte Chek Parity  - 0/1/2 equals NoCheck(8N1)/EvenParity(8E1)/OddParity(8O1)

Write example: w 434000 3 9 3 0 is...
      434,000 MHz 9600 baud in air, 20mW, 9600baud on UART, No Parity(8N1)
After 30 seconds with no keyboard input we will emit a char every two second
  
```

Figure 13: Configuration of the transceiver

If you type 'r' and hit enter, the program will return the current configuration for the transceiver. To reconfigure the radio, type 'w' and the 6 parameters needed, with space between each parameter.

Note that you have to configure both transceivers with the same settings to be able to use them together.

RF-Magic

The transceivers can also be programmed by using RF-Magic, however the program has a lot of bugs, and we don't recommend using this program. Your hair will turn grey!

Once the drivers are correctly installed and the hardware is ready to use, start the interface program RF-Magic as administrator. Most likely during start, an error will be generated. This is because the program cannot find the device on the expected com-port.

If you have trouble connecting the USB-UART bridge with RF-Magic, check that you have it in the *Device Manager* and try to set the com-port number as low as possible.

When RF-Magic has found the device it will appear in the bottom of the window as shown in Figure 15.

Still not working? Try restarting the computer. If the problem consists; uninstall the driver and reinstall it.

When you have the RF-Magic up and running you can configure the radio to your designated frequency and bitrate. Remember to configure both radios, otherwise you won't be able to communicate between the CanSat and ground station.

For your CanSat you have been given a pre-set frequency. If not, a list of frequencies are listed in the

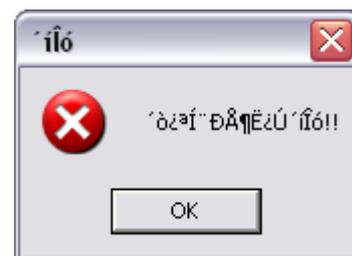


Figure 14: Error Message

Appendix. The bitrate should be set to 9600bps.

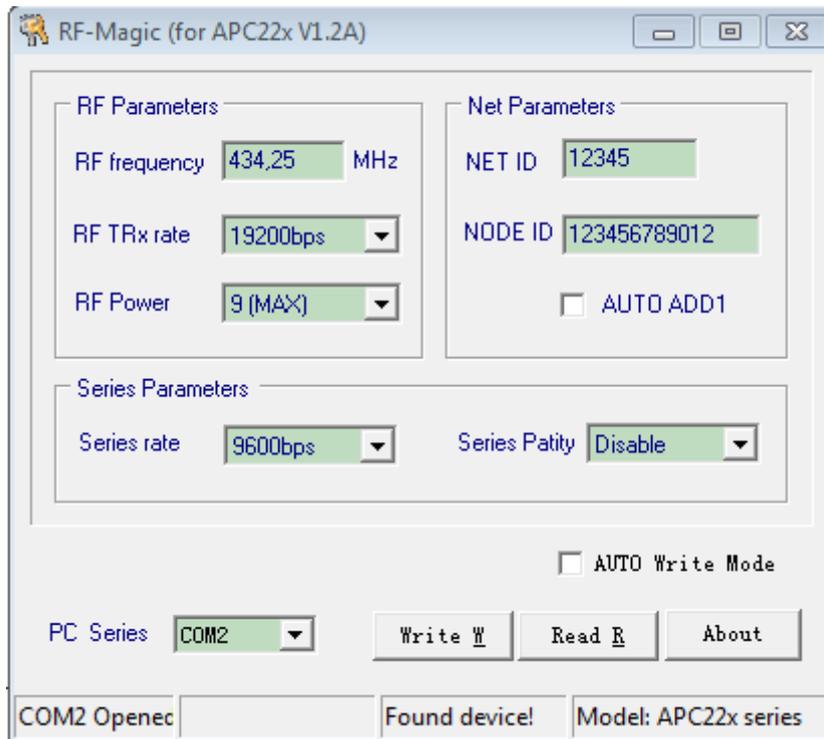


Figure 15: RF-Magic with working COM-port

Preparing CanSat Shield Radio

On the CanSat shield, **Feil! Fant ikke referansebildet.** there are two sets of jumper pins, **J2** and **J3**. When **J2** and **J3** is not connected you can program the Arduino. By setting the jumpers at **J2-1** and **J2-2** you can transmit data to the radio. When placing jumpers at **J3-1** and **J3-2** you enable data storage in the SD-card (if it's connected).

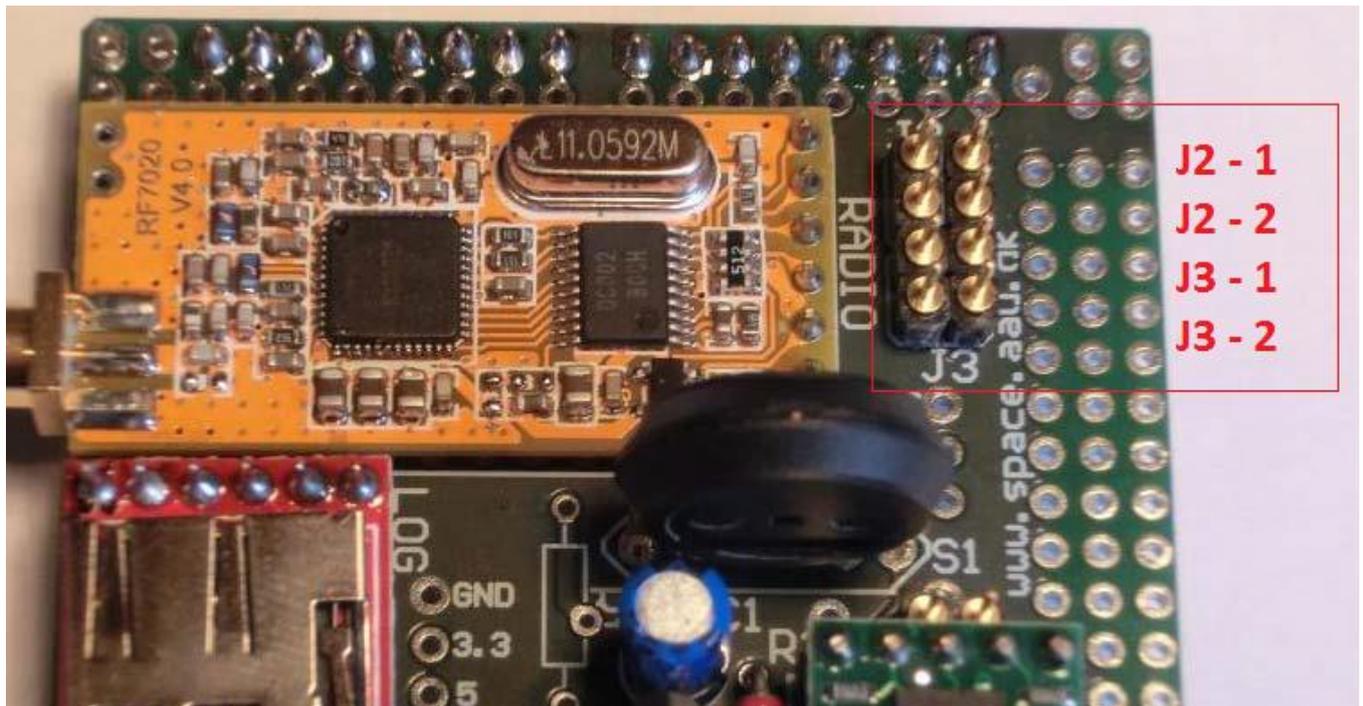


Figure 16: The CanSat Shield with radio

Testing the radio

When you are finished programming the CanSat kit you should test that the radio link is working properly. Place jumpers on **J2**, connect the receiver to the PC via the USB to UART Bridge (USB-converter) and open the Arduino software.

Change the COM-port from Arduino Uno R3 (COMxx) to Silicon Labs CP210x USB to UART Bridge (COMxx) in the Arduino software by going to **Tools → Serial Port**. If you don't remember which COM-port that is the correct one, go to the **Device Manager**.

Press the serial monitor to look at the data that is coming from the radio. To store data we use *Tera Term VT*.

Tera Term VT

Tera Term VT is a freeware that reads and store data coming from the serial ports. You can download the latest version from logmett.com.

When you have installed the program, run it.

Then you will get a window to create a new connection. Select **Serial** and then the COM-port you want to fetch data from. Then press **OK**.

If you don't get any data, bend the Enable pin on the transceiver, which is labelled **EN**. Be careful not to brake the pin off. This pin will be needed when you program the transceivers.

If you want to store the data you have to press **File → Log**, and mark **Append** to let new data append to the existing file.

Mark **Plain text** if the data shall be interpreted as ASCII-signs.

In addition to the terminal window, another window will also appear that lets you know where the file is stored and the size of that file. You can control when to record data by pushing the **Pause\Start** button.

If the data is plain numbers with commas between it is easy to copy into Excel or other mathematical programs.

If you want to change the baud rate or select another comport, this can be done by going to **Setup → Serial port**.

Parachute design

Satellites normally do not return to Earth in a parachute. At the end of their useful life, a satellite will be put in a different orbit. For satellites orbiting at a low altitude this could mean they will burn up in the atmosphere. Satellites further away will end up in a much more distant parking orbit and will circle our planet forever. Sometimes however the spacecraft has to return to earth with samples or astronauts. One of the solutions is then to descend in a parachute.

When the CanSat is deployed it must have a device to slow it down, otherwise it will crash into the surface. The parachute also helps ensure that the CanSat stays in an upright position. This is particularly important because it helps to maintain proper antenna orientation, which maximises the chances of receiving telemetry. This lesson will guide you through the different steps needed to design and built your parachute.

Required Descent Parameters

The values are still preliminary; they may change in the future. However they can be used as a temporary guideline. Uncertainties in the Launch Campaign could lead to a different value for the eventual descent velocities.

Minimal descent Velocity: 8 m/s

Maximal descent Velocity: 11 m/s

Maximum allowed mass: 350 grams

Drag coefficients:

Semi Spherical: 1.5

Cross Shaped: 0.8

Parapent: depends on the design (can be determined by tests)

Flat, hexagon: 0.8

Parachute production

When the design of the parachute is finished you can start the production process. There are however a few important issues to keep in mind during this process. Deployment of the parachute will be relatively violent, so the fabric and fibres you use need to be strong. Most often you can get nylon cord and ripstop fabric at a kite shop. These materials are ideally suited for the parachute.

When cutting the fabric keep in mind that some of the fabrics need to be cast double in order to sew it.

More handy tips on parachute production can be found here:

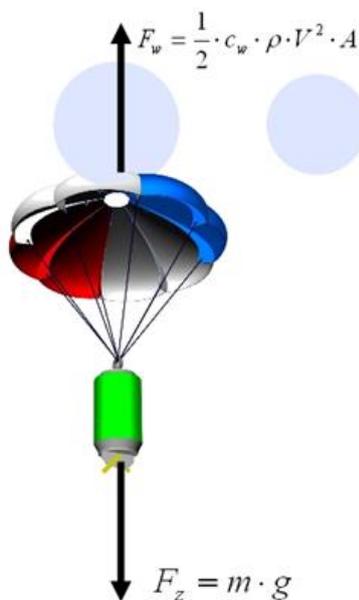
<http://www.nakka-rocketry.net/paracon.html>

When the parachute is done, the best way to check if it works is to actually test it.

Example Assignments

The following assignments can be performed when working on the parachute.

- Calculate the impact speed of the can without a parachute (when released from 1 kilometre altitude).
- Calculate the minimum required area for your parachute when you use a cross parachute. What size should the squares be on the chute?
- Perform the same calculation for a spherical parachute. What is the radius?
- Test the descent velocity of your parachute with a soda can?



- Try out different solutions for the parachute. A parachute with some holes in it or perhaps multiple small parachutes? Both will enhance the stability of the CanSat.

Descent Physics

Before we can start making the parachute we will have to figure out how big it should be. More specifically, we need to calculate how much surface area the parachute will need in order to fulfil the requirements.

Logic suggests that the bigger the parachute the slower the object's descent velocity. Later on this principle is shown with some basic equations. Although it would be very beneficial for the CanSat to have a very low descent rate, a limit has been set to ensure that the CanSat will land near the launch area. If the descent rate is too slow the CanSat may drift kilometres away along with the wind, which is neither allowed nor desired. For safety reasons there has also been set a maximum descent rate.

To design the parachute we'll use some simple physics. We use a simplified model to estimate the area of the parachute, after which we can start on the construction.

During the descent two forces will be acting on the CanSat. Gravity will pull on the can and accelerate it towards the ground, and the drag force on the parachute will pull the CanSat in the opposite direction and slow down the descent rate. The two forces are shown in the image above.

When the CanSat is deployed, the force of gravity will cause it to accelerate. After a few seconds the drag force from the parachute will reach equilibrium with the force of gravity. From that point on, the acceleration will be zero and the CanSat will descend at a constant velocity. This constant velocity has to be greater than the minimum descent velocity specified in the requirements. For the following calculations we can use this minimum value as the constant velocity of the CanSat.

The gravity force is equal to:

$$F_g = m * g$$

In this equation:

m: Mass of the CanSat

g: Acceleration of gravity, equal to $9.81 \frac{m}{s^2}$

The drag force of the parachute is equal to:

$$F_D = 0.5 * C_D * \rho * A * V^2$$

In this equation:

A: Total area of the parachute (not just the frontal area)

C_D : Drag coefficient of the parachute. This value depends on the shape of the parachute.

ρ : Local density of the air, assumed to be constant at $1.225 \frac{kg}{m^3}$.

V: Descent velocity of the CanSat

Given a desired velocity, you can easily rewrite these equations to calculate the area needed for the parachute.

Semi-spherical Parachute Design

A semi-spherical appearance is the most common shape of a parachute. Although it is not hard to make one, it can be quite time-consuming to get the right shape. The figure below should help out.

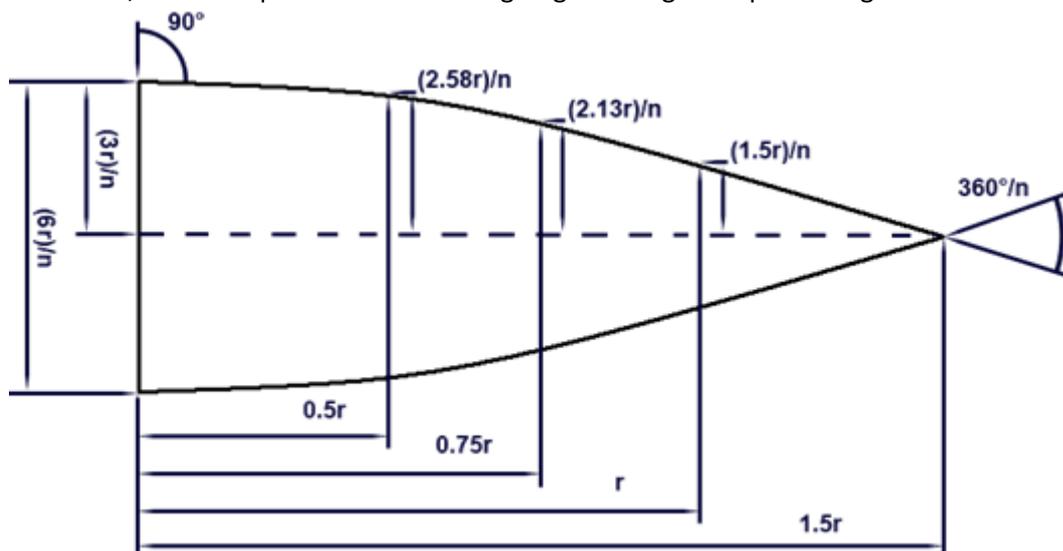


Figure 17: Semi-spherical parachute

n stands for the number of needed parts

r stands for the radius of the parachute.

Cross Parachute Design



Figure 18: Cross Parachute

Instead of using a semi spherical shaped parachute you can also choose a cross shape. The advantage of this shape is that it's easy to make. If you want to know more about cross shaped parachutes you can check the following link:

<http://www.nakka-rocketry.net/xchute1.html>

Parapent

A parapent shaped parachute acts a bit like a wing. Because of its shape you can use it to steer. The design of a parapent is more complex than that of the other shapes mentioned. You will have to do some more research if you wish to use this type of parachute.



Figure 19: Parapent

Flat parachute design

Most commonly available parachutes are in fact created from standard two-dimensional flat geometric figures, such as hexagons or octagons.

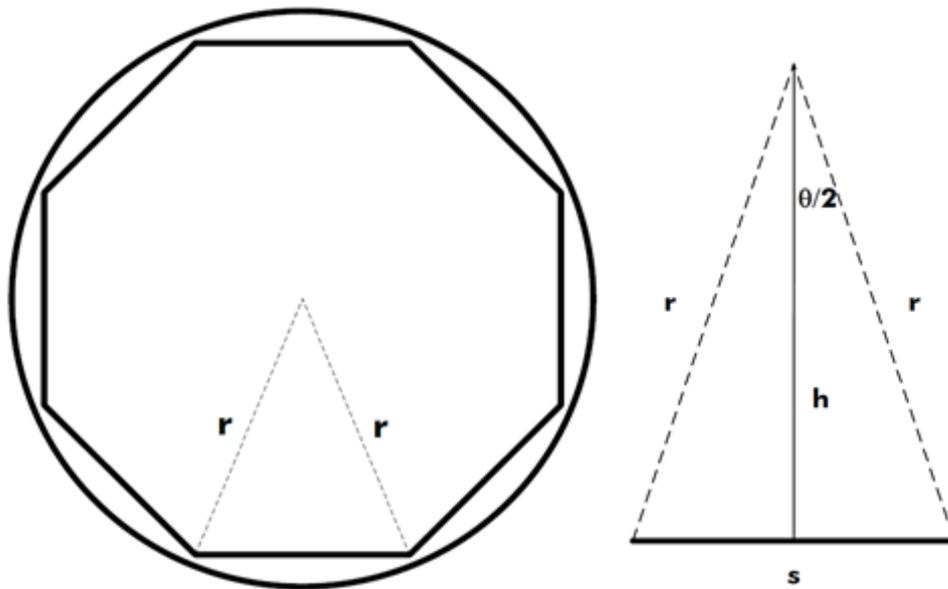


Figure 20: Octagon parachute

From Figure 20 above it is shown that the parachute consist of 8 equal triangles. Hence the total area for the parachute would be $A = 8 * AT$, where the area of one triangle is $AT = s * \frac{h}{2}$. By combining these two equations we get $A = \frac{8*s*h}{2}$. You can read more about how to calculate the area of a flat parachute here:

<http://www.sunward1.com/imagespara/The%20Mathematics%20of%20Parachutes%28Rev2%29.pdf>

As soon as you know the total area (A) together with the drag coefficient (Cd) you can easily determine the descent rate for your CanSat. If you do not know the drag coefficient for your parachute, you can do drop tests of the CanSat to find the terminal velocity.

CanSat Design

Introduction

The whole concept of CanSat is that neither weight nor volume shall be bigger than a standard soda can. Electronics must be mounted on a robust bracket that handles the stress of a launch and separation from a rocket. The parachute must be connected to the bracket, not the can! The can is just a shield to protect the electronics against direct impact.

The antenna should be a tread antenna because of its flexibility so when the CanSat lands, the antenna won't be damaged during impact.

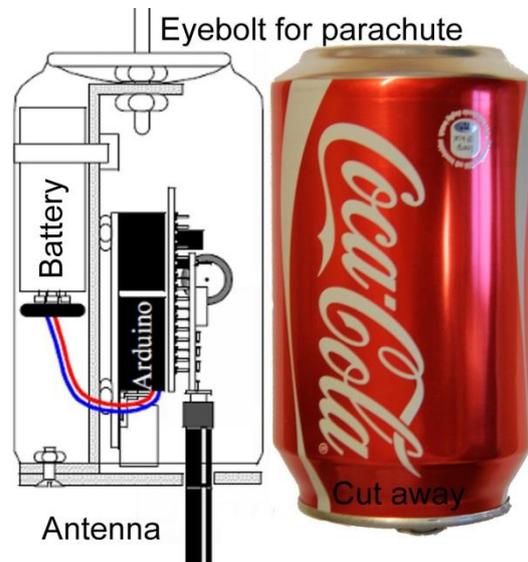


Figure 21: The CanSat with bracket

Competition requirements

NAROM has since 2009 held national CanSat competitions and also two European competitions in cooperation with the European Space Agency.

NAROM uses the following set of requirements that are compatible with the Intruder rocket.

- All the components of the CanSat, with the exception of parachute, antenna and GPS, must fit inside a European soda can: 115 mm high and 66 mm diameter.
- The maximum mass of the CanSat is limited to 350 grams.
- The CanSat should have a recovery system, such as a parachute.
- The deployable subsystems and recovery system can exceed the length of the primary structure, but can't be deployed before the CanSat is fully ejected from the rocket.
- Flight time is limited to 120 sec (If not other agreements have been made).
- The descent rate must be between 8 m/s and 11m/s (If not other agreements have been made).
- Explosives, detonators, pyrotechnics, flammable materials, dangerous materials and biological payloads are strictly forbidden. All materials used must be safe for personnel, equipment and the environment. Material Safety Data Sheets (MSDS) will be requested in case of doubt.
- The CanSat shall operate with a battery or solar panels. It must be possible for the systems to be powered on for three continuous hours.
- The CanSat must be able to withstand an acceleration of up to 20G.
- The battery must be easily accessible, in case it has to be replaced or recharged in the field.
- The total budget of the CanSat should not exceed €500.

Note that these requirements are just temporary guidelines. In the event of a competition the requirements will be announced.

The Bracket

There are several ideas on how the brackets should look like. We are going to make suggestion on how it can fit the Arduino board.

The drawing, Figure 2222, shows the bracket with dimentions and placements of holes before it is bent. The material you can use for the plate is 0.5-1 mm aluminium.

The plate is bent 90° inwards on top and 90° outwards on the bottom. Where to bend the plate is shown on the drawing.

Remember holes for strips to fasten the battery.

To make the CanSat easy to turn on and off, a switch is recommended to be placed between the battery and the CanSat kit. But remember that the switch will be a weak link during launch campaign.

In Figure 23 we can see an example of how the CanSat could look like.

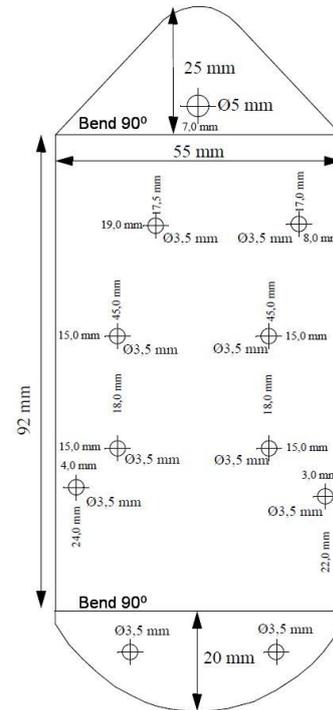


Figure 22: CanSat Bracket design



Figure 23: Finished CanSat

Antenna design

The antenna that is included with the APC220 radio is a 433MHz Rubber Duck antenna. This antenna is robust and great for testing, but won't fit inside a soda can.



Figure 24: Duck antenna

A good alternative is the simple thread antennas that can be soldered directly on to the transmitter output or attached to a SMA-connector. Normally such antennas will be a quarter-wavelength. The thickness is not critical; the most important is the flexibility and durability.

Antenna length can be calculated from equation below when the frequency (f), and velocity of light (c) is known:

If we have a frequency of 434 MHz, then the equation for calculating the length of a quarter wavelength is:

$$L = \frac{c}{4f} = \frac{(3 * 10^8)}{(4 * 434 * 10^6)} = 0.173m$$

By this calculation we find that the antenna should be 17.3 cm long. We can build this antenna by using a coaxial cable. Remove 17.3 cm of the plastic jacket and the metallic shield from the cable, leaving the centre core and dielectric insulator. Make sure that the shielded part of the cable reaches all the way out of the can before it is stripped.

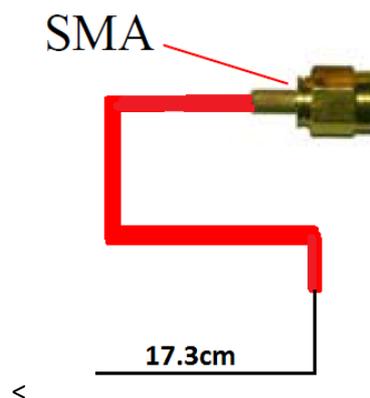


Figure 25: Thread antenna

Appendix

Frequencies

Frequencies (MHz)	Team	Frequencies (MHz)	Team
433,050		433,950	
433,100		434,000	
433,150		434,050	
433,200		434,100	
433,250		434,150	
433,300		434,200	
433,350		434,250	
433,400		434,300	
433,450		434,350	
433,500		434,400	
433,550		434,450	
433,600		434,500	
433,650		434,550	
433,700		434,600	
433,750		434,650	
433,800		434,700	
433,850		434,750	
433,900		434,800	

Component list

Compoent	Dealer	Price (NOK) -tax	Price (NOK) +tax	Link
Arduino Uno R3	Robonor	184,00	230,00	http://www.robonor.no/mage/default/arduino-uno-r3.html
Radio - APC220	Robonor	359,20	449,00	http://www.robonor.no/mage/default/apc220-radio-communication-module.html
Logger - OpenLog	Robonor	183,20	229,00	http://www.robonor.no/mage/default/openlog-datalogger.html
LM35DZ (Temperatur)	Elfa	10,54	13,18	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=73-276-79&toc=19595
Capacitor 1uF	Elfa	1,57	1,96	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=67-043-16&toc=19089
Resistor 75 ohm	Elfa	0,71	0,89	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=60-591-86&toc=19848
Resistor 10 kohm	Elfa	0,71	0,89	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=60-592-39&toc=19848
Breakaway headers	Elfa	12,70	15,88	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=43-839-21&toc=19754
Jumper 2.54mm 2p	Elfa	1,98	2,48	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=43-711-07&toc=19754
Female Headers 7p	Elfa	7,58	9,48	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=43-783-03&toc=19754
Female Headers 2p	Elfa	4,45	5,56	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=43-782-53&toc=19754
Battery Connector	Elfa	8,98	11,23	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=42-043-01&toc=19108&q=batterih%C3%A5llare+9v
USB2.0 Cable A/B	Elfa	24,20	30,25	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?init=2&shop=ELFA_NO-NO&item=25-112-42
MicroSD Card 2GB	Elfa	22,80	28,50	https://www.elfaelektronikk.no/elfa3~no/no/elfa/init.do?item=73-019-80&toc=0&q=sandisk
MPX4115A (Trykksensor)	Farnell	83,97	104,96	http://no.farnell.com/freescale-semiconductor/mpx4115a/sensor-abs-

				press-16-7-psi-867-6/dp/1457150?ref=lookahead
NTCLE100E3103JB0 (Temp)	Farnell	5,06	6,33	http://no.farnell.com/vishay-bc-components/ntcle100e3103jb0/thermist-or-ntc-10kohm-radial/dp/1187031
Coax Cable RG316	Farnell	173,45	216,81	http://no.farnell.com/jsp/search/product_detail.jsp?sku=1776766
Parachute 18 inch Nylon	Apogeerockets			http://www.apogeerockets.com/Building_Supplies/Parachutes_Recovery_Equipment/Parachutes/Low_Power/18_Nylon_Parachute
MMA7361L (Accelerometer)	Pololu.com			http://www.pololu.com/catalog/product/1246
Aalborg CanSat shield	Aalborg Uni.			http://www.control.aau.dk/~jdn/edu/cansat-kit/shield.html
Eyebolt 5mm 5pack	Biltema		14,90	http://www.biltema.no/no/Bygg/Beslag/Krok/Oyeskruer-5-pk-86254/

Simulation program “Launch”

There is some software available to do a drop test simulation of a CanSat. There’s a non-commercial software available called “Launch” that can be used do a simple simulation. The process on how to do this is shown below. Consult your CanSat instructor to get the software Launch and required data file “Intruder.ldf”.

When you start the program “Launch”, choose “Open” under the “File” menu. From here you have to navigate to the folder called “PROJECTS” and select the file “Intruder.ldf”:

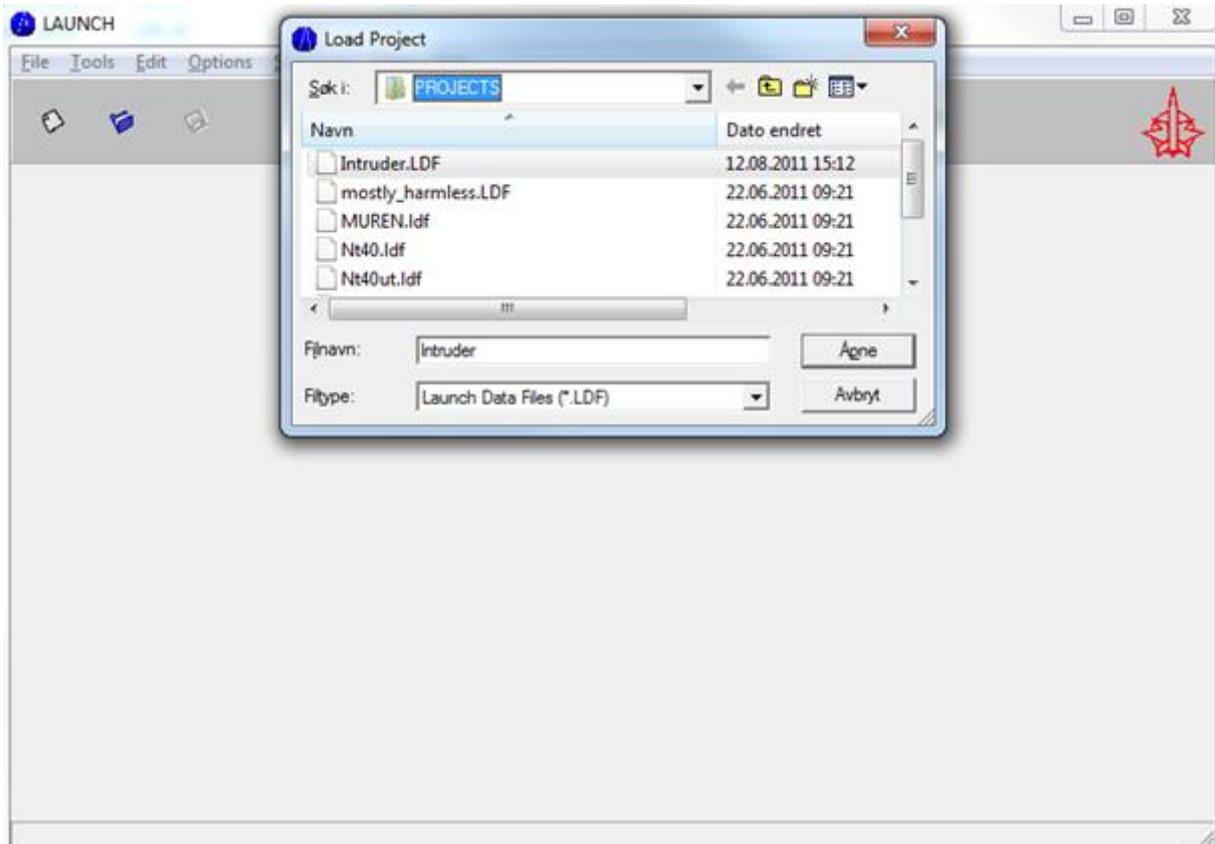


Figure 26: Open the file "Intruder.LDF"

You will then get a new window which shows several flight events for the Intruder rocket that NAROM is using to launch CanSat. For this simulation we will only do minor adjustments of the parameters under “initial settings”. Click on “Initial settings”:

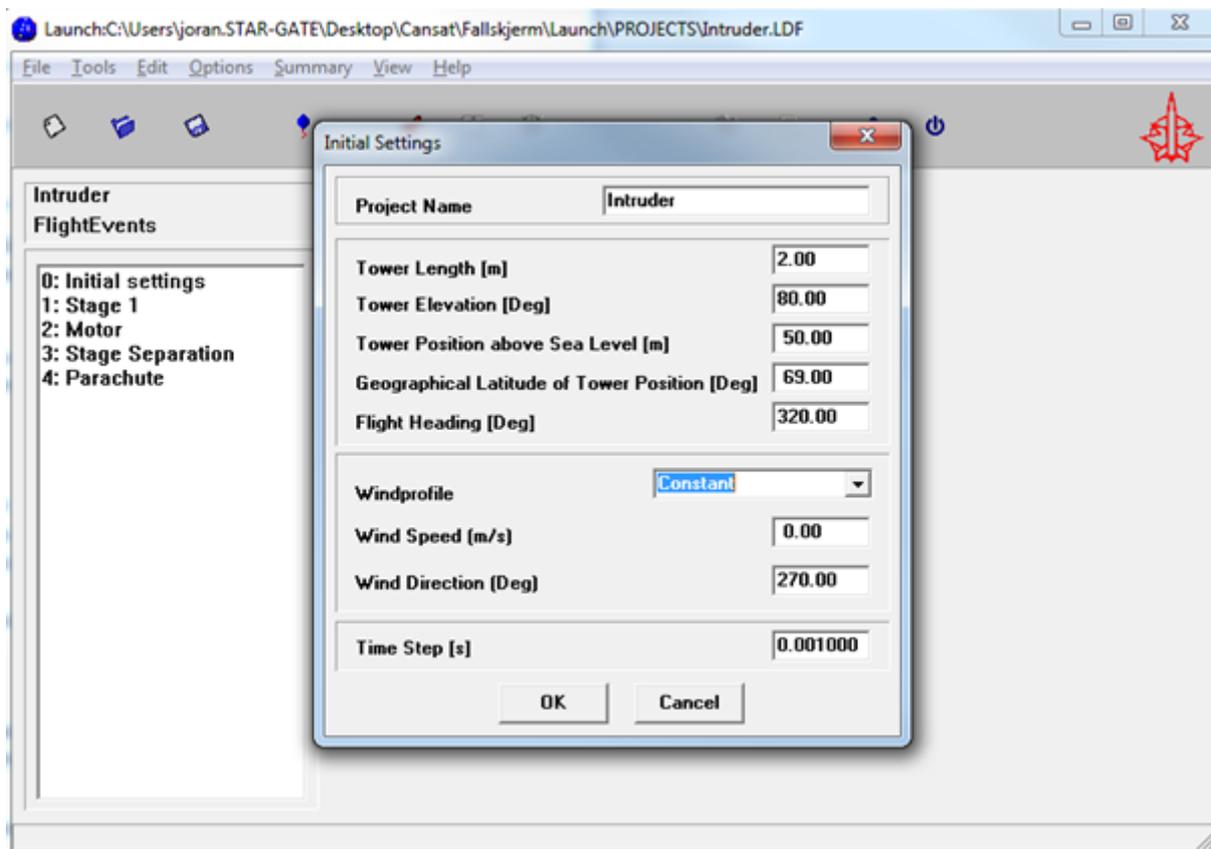


Figure 27: Initial Settings

From “Initial settings” you can adjust the placement of the launch rail. In Figure 27, data for the launch site at NAROM has been inserted. In addition it is possible to add a wind profile, wind speed and wind direction to see what effect this will have on the descent of your CanSat. At the bottom you will find the time step for the simulation. You do not need to change the time step for this simple simulation. As soon as you have entered all your inputs, press ok.

The next thing on the flight event list is the “Stage 1” and “Motor” flight event. These two flight events have been set by NAROM to model the Intruder rocket. You do not need to change any of these. You can now proceed and click on the flight event “Stage Separation”:

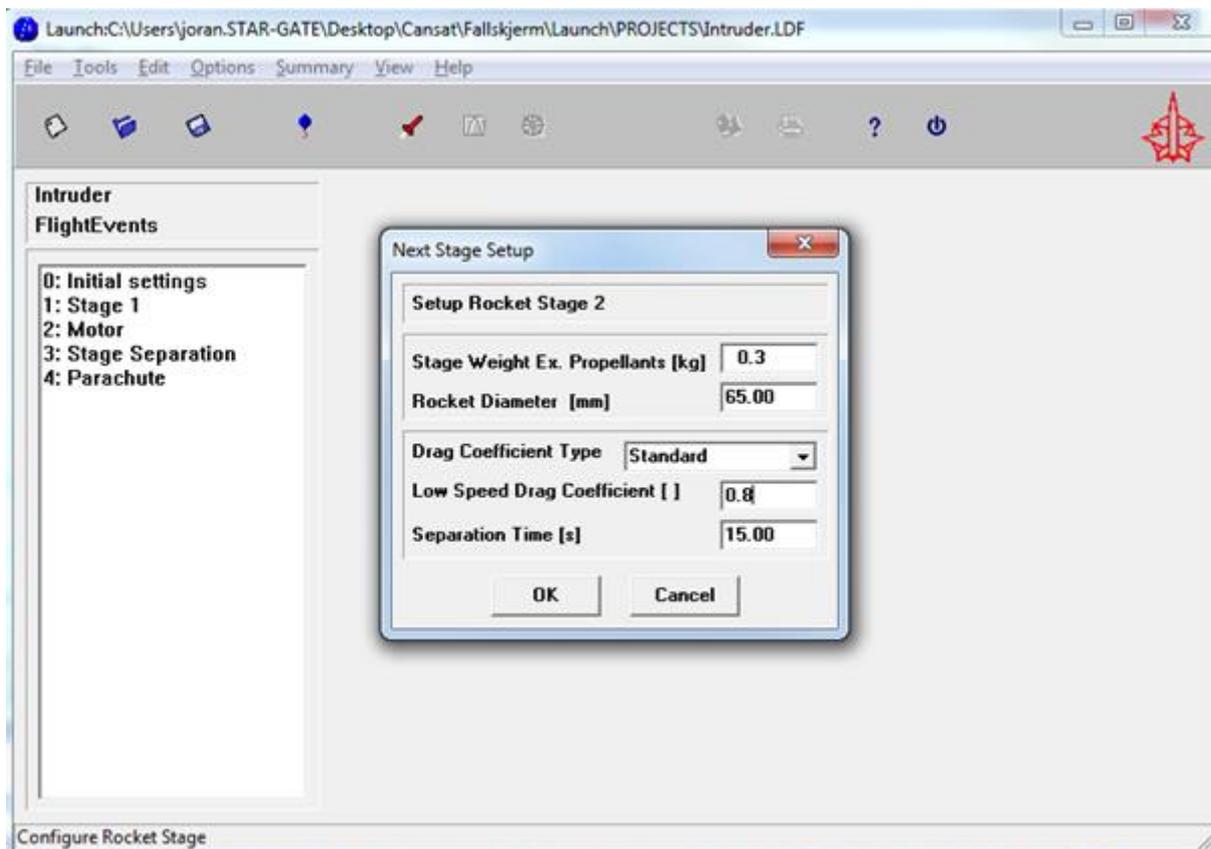


Figure 28: Stage Separation

Under “Stage Separation” you will have to enter the parameters for your CanSat including the weight, diameter and drag coefficient. You can keep the separation time at 15 seconds which is just after apogee of the Intruder rocket.

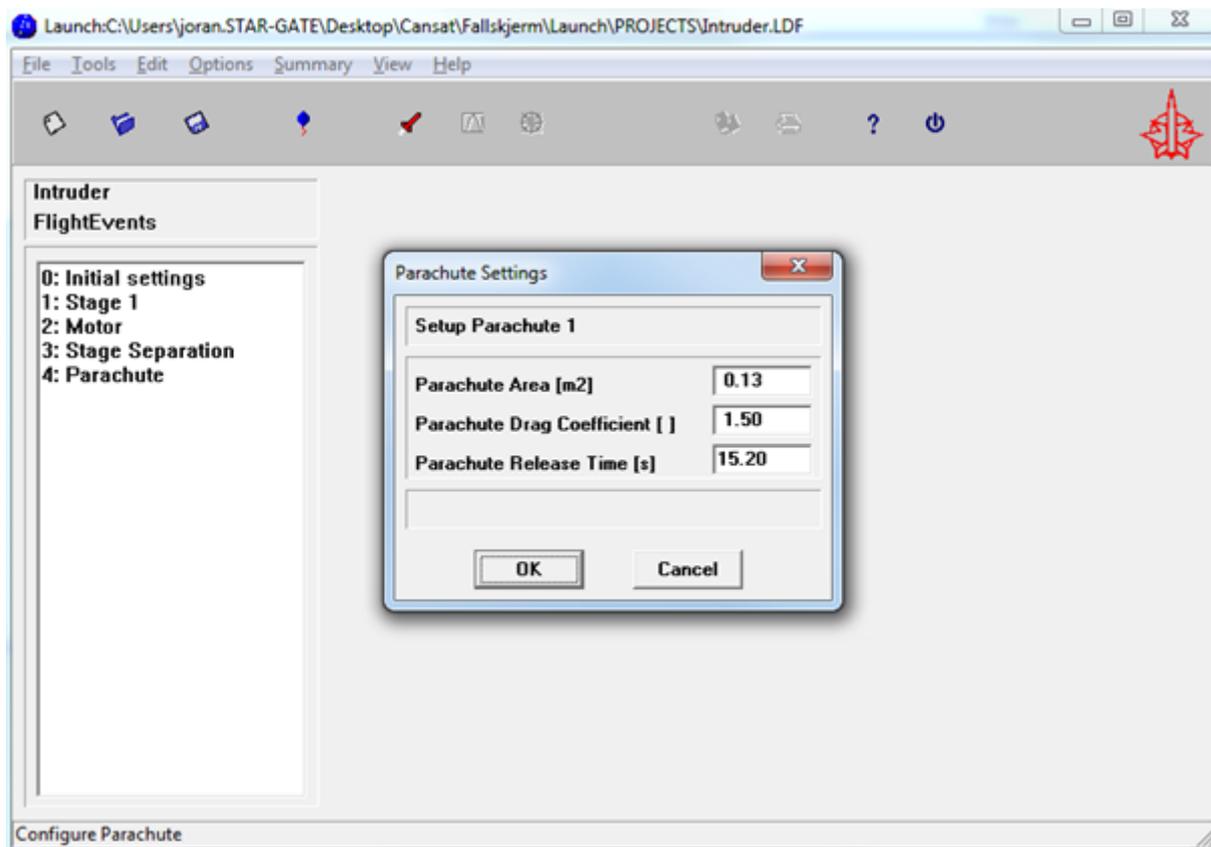


Figure 29: Parachute

Under “Parachute settings” you will have to enter the total area of the parachute and the drag coefficient of the parachute. We assume that the parachute deploys just after the release of the CanSat from the Intruder rocket and therefore set the “Parachute Release Time” to 15.2 seconds.

You are now ready to perform the simulation based on your data input by clicking on the red symbol shaped like a rocket. You will then get the following window:

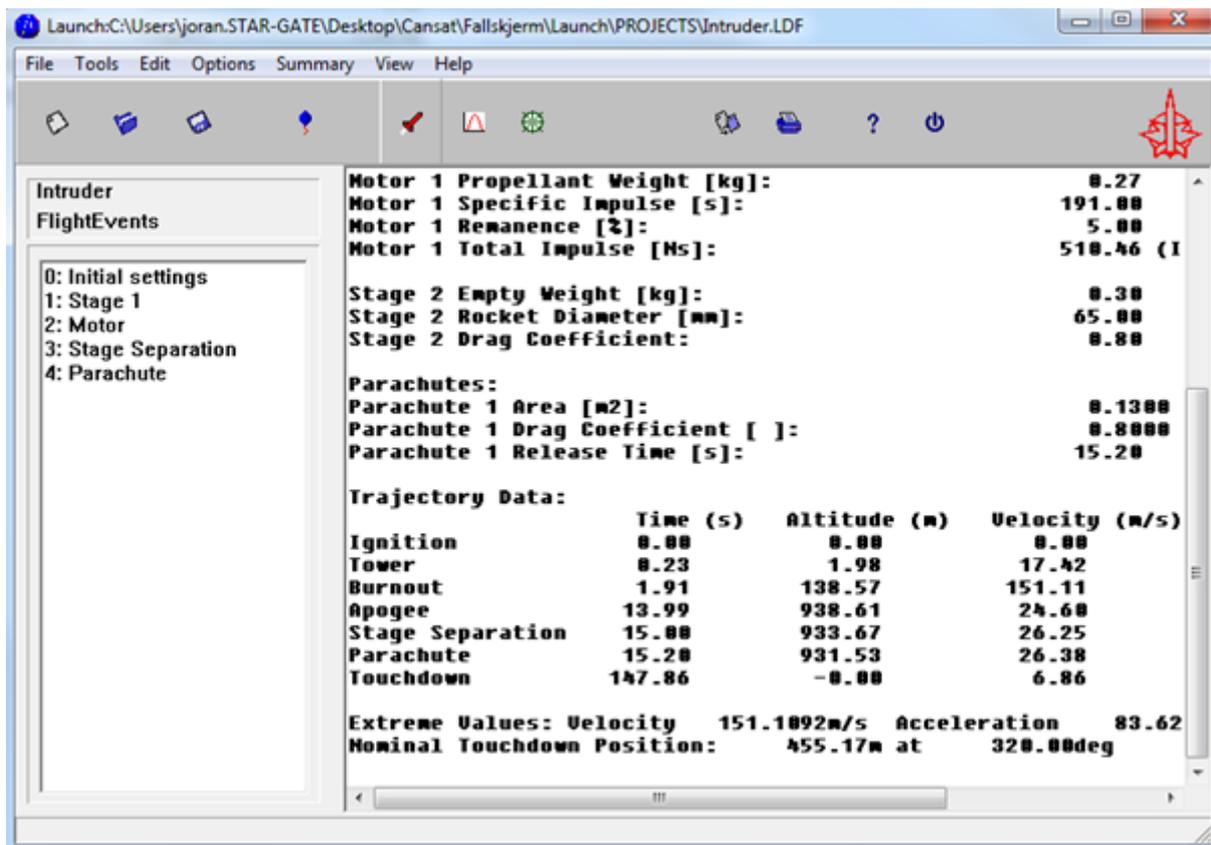


Figure 30: Run Calculation

From the window you can do a readout of some of the calculated parameters, such as “Touchdown” which in this case is 6.86 m/s . This will then be the terminal velocity of your CanSat. You can get additional information by clicking on “Trajectory plot” next to the “Run calculation”.

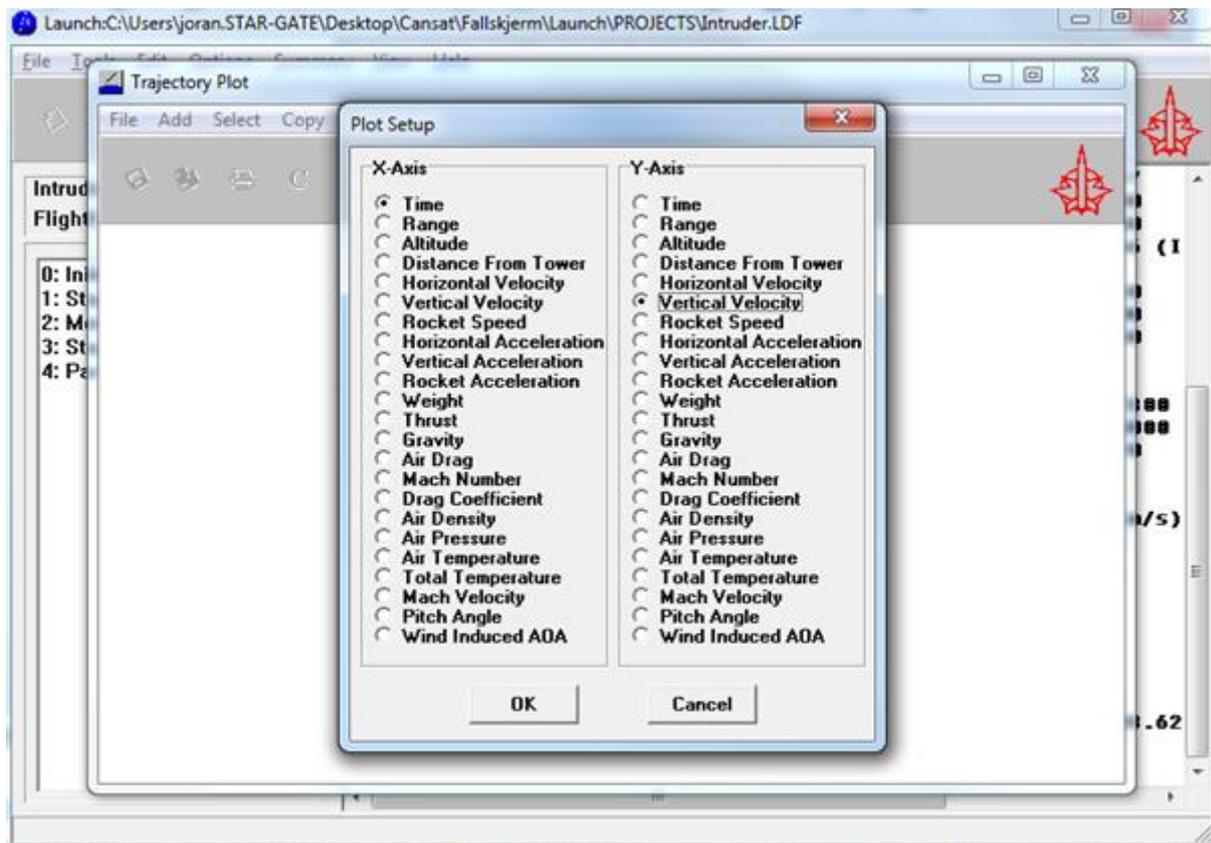


Figure 31: Trajectory plot

From the “Trajectory plot” you have the option of selecting several different plots. You would probably be most interested in the vertical speed vs. time:

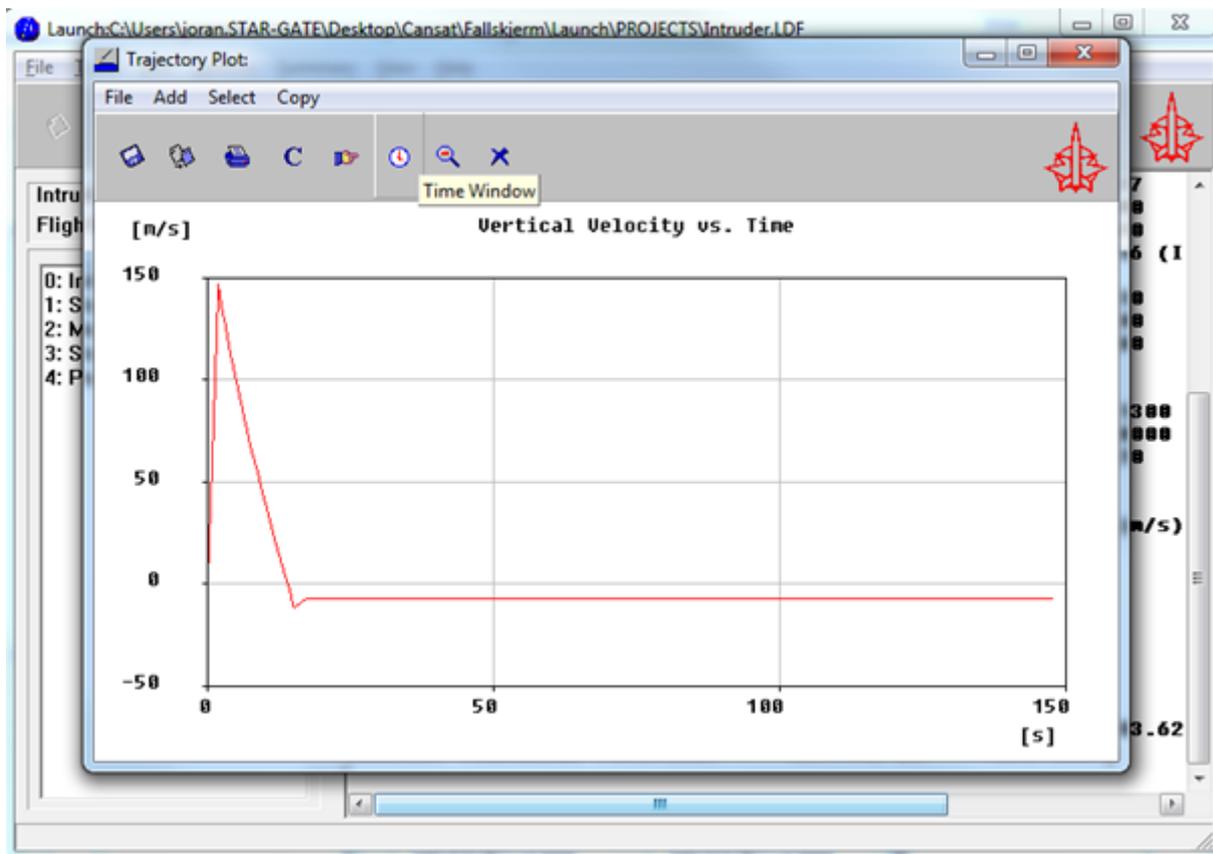


Figure 32: Vertical speed vs. time

The first 15 seconds shows the speed of the rocket whereas the deployment of the parachute is at 15.2 seconds. You can zoom in on the different areas. It is also possible to export data from the plot to a text file if you for instance want to import the data into Excel. To export the data, select “Export data” from the “File” menu.