Manual for scientific instruments on the NAROM student rocket campaign

*Note: this manual will only make sense when the reader has a fairly good overview of how the NAROM encoder work.*

There are numerous sensors on the NAROM student rocket, some of which are analog and some are digital. The digital sensors are more “modern”, but the analog sensors are good for educational purposes and has certain parameters that you cannot find (affordably) on digital ones: e.g. the bandwidth of the magnetic field sensors and the huge span of the accelerometer. A mix between the two will fit the student rocket very well (remember that the acceleration are high and the spin is fast).

In this document we start by giving an overview of the sensors, then some background information of the differences between analog and digital sensors are given, before going more in-depth on each of the scientific sensors that we will make and use. All sensors will have a short description, schematics and parts list, and any information that is needed to build and use the sensor board.

Note that the term “sensor” is used for the measuring IC itself (often a black chip) and that “sensor board” is used for each PCB with electronics on it, that being the sensor and any other electronics needed.

# Contents

[Contents 1](#_Toc534964570)

[1. Brief overview of the sensors 2](#_Toc534964571)

[2. Difference of analog and digital sensors 2](#_Toc534964572)

[3. Details of the digital sensors 3](#_Toc534964573)

[3.1. Rocket encoder interface 3](#_Toc534964574)

[3.2. The Teensy LC development board 4](#_Toc534964575)

[4. In-depth description of sensor boards 4](#_Toc534964576)

[4.1. Analog accelerometer (MMA3202) 4](#_Toc534964577)

[4.2. Magnetometer (KMZ51) 6](#_Toc534964578)

[4.3. Pressure (MPX5100AP) 7](#_Toc534964579)

[4.4. NTC Temperature array 9](#_Toc534964580)

[4.4.1. Using the NTC resistor as a sensor 9](#_Toc534964581)

[4.4.2. The temperature array vertical frame 11](#_Toc534964582)

[4.4.3. Converting from ADC raw values to voltage 12](#_Toc534964583)

[4.5. IMU (LSM9DS1) 13](#_Toc534964584)

[4.5.1. The IMU and GPS vertical frame 13](#_Toc534964585)

[4.6. GPS sensor (Skytraq Venus 638) 15](#_Toc534964586)

[4.7. PTU (weather balloon sonde) 17](#_Toc534964587)

[20](#_Toc534964588)

[5. NAROM PTU template 20](#_Toc534964589)

# Brief overview of the sensors

This document will introduce the various scientific instruments (there are others sensors that are monitoring the payload).

The analog sensors are:

* Accelerometer parallel (x axis) and perpendicular (y axis) to the axis of travel
* Magnetometer (y axis)
* Pressure sensor

The temperature and light sensor is not considered a scientific payload, but they are also analog.

The digital sensors are:

* Temperature array
* IMU (3 axis accelerometer, gyroscope and magnetometer)
* GPS

When we launch a rocket, it is important to have data about the state of the atmosphere, so two weather balloons are also launched (in addition to the two student balloon release, you will also get data from the ASC balloon release that are done one hour prior to the launch). For each student balloon release, some of the students will also make a weather balloon sonde (called a PTU sonde). This contains these sensors:

* Absolute pressure
* GPS
* Temperature

# Difference of analog and digital sensors

*(Consider this section as background information).*

One can ask what the difference is between the digital and analog sensors. Almost all sensors are inherently analog in nature, since nature is not digital at all. All of the sensors on the rocket and weather balloon on the student campaign is indeed analog, but the distinction we make here is how the information from the sensors are transferred to the encoder/main board.

Let us take measuring the acceleration as an example. On the student rocket two different sensors measure the acceleration: an analog and a digital (on the IMU). The analog accelerometer measures in two directions with a huge range: in the axis of travel and perpendicular to it (in the plane of the sensor bracket), and the digital sensor has a range of in all the three measuring direction, which is less than the maximum acceleration during the rocket thrust phase.

Both sensors will actually work in quite the same manner: in each measuring axis a very tiny MEMS silicon mechanism will respond to acceleration in a certain way, often changing its capacitance. This capacitance can be converted to a voltage (or how a voltage is changing) using clever techniques.

Here the similarity ends between the two sensors:

* On the “analog” sensor capacitance is converted to a voltage, amplified, filtered and finally compensated for temperature before the voltage is set as an output on one of the sensor pins, and it is up to the user to use this information either further in the analog domain or converted to the digital domain.
* The digital sensor, however, has a small embedded digital “computer” that converts this capacitance to the digital domain (no information is given to how this is done) by a. analog to digital converter (ADC), filtered and stored on-chip on the embedded memory. On this digital part, there is also a serial, digital interface (both I2C and SPI, specifically). In our case, we use the I2C interface, and the “master” (normally a microcontroller, which it is on our sensor boards) reads the measurements made by the ADC. Also, since the sensor is “smart”, the user can decide certain parameters, with the most important ones the sampling rate, bandwidth (how high or low frequencies one can measure) and absolute measurement range ( and in our case).

Although the digital sensors are more complex, in the modern world we live in, the digital sensors is widely used and made in so huge numbers that they are often much, much cheaper than using only analog sensors.

# Details of the digital sensors

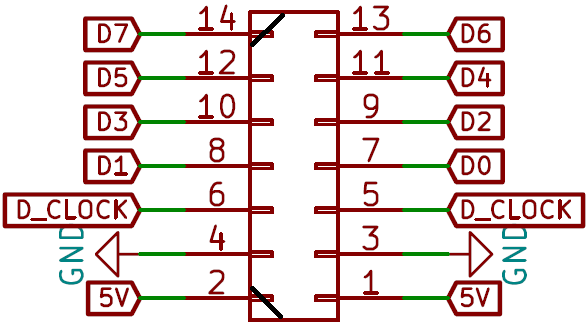
As discussed in the previous section, the digital sensors has only a tiny fraction of all of its “workings” in the analog domain with the rest in the digital domain. The encoder only read a custom digital interface and cannot work with the more well-known interfaces like I2C, SPI and UART, which is most commonly used by sensors as the ones we’re going to work with. Hence, we need a controller that read the sensor data and pack these in a specific way and send this to the encoder on the custom encoder interface, and a normal microcontroller is chosen for this purpose. This is described in-depth in this section.

The interfaces used on some of the digital sensors here is only I2C (the IMU) and GPS (UART) is not further discussed here. These are well-known interfaces and huge amount of documentation and tutorial exist, and the reader is referred to these (Wikipedia is a great resources for this).

## Rocket encoder interface

The encoder is an Atmel Atmega 328P (the same as on the famous Arduino Uno development board), which is not very powerful (it runs on 16 MHz). To minimize time spend on reading data from the digital sensors, a parallel interface is chosen. The microcontroller only use 1 clock cycle to read each sensor.

The physical connector on the encoder interface has a pinout given as:



The connector is of the type Micronector/Datamate 2x7 with polarity, note the two black lines that mechanically unables the connector to connect the the board except in the correct polarity. Pin 1 and 2 is power, 3 and 4 is ground, 5 and 6 is a clock signal from the encoder to the sensor board and 7 to 14 is the data from the sensor to the encoder, with D0 is the most significant bit (often called MSb) and D7 the least significant bit (LSb).

For each new byte that is transmitted (one byte per encoder frame) the encoder first set the D\_Clock signal (also called the digital clock signal) is first set to 5V (logical high) for 64 microseconds before going low, and the port is sampled by the encoder 64 microseconds after the signal going low again. That is, the encoder samples the parallel data port 128 microseconds after going high. Note that the sensor board do not have any information of anything that happens on the encoder except the clock signal. This means that we cannot sync the data according to the frame counter or any other information, and hence we need to come up with our own format of the data. This format is discussed in the sections for each sensor.

## The Teensy LC development board

As already discussed, we need something between all the sensors and the encoder. The temperature array sensor board (discussed later) samples analog sensors, and the other digital sensor boards uses digital sensors and works more as a buffer. What they all have in common is that a microcontroller gathers all the information that we will send to the encoder, stacks them in a given format and send the information to the encoder through the custom interface.

For this job we have chosen is the Teensy LC development board, based on a Freescale (now NXP) Kinetis KL26 Cortex M0+ microcontroller running at 48 MHz. This is an affordable, simple to use and low-power microcontroller board which is perfect for our needs. To keep it student friendly, all the programming is done using the Arduino IDE, and all the pre-made code is available for the students to read, copy and used later if needed.

More details on the board and its capabilities, pinout, schematics and more is found here:

<https://www.pjrc.com/teensy/teensyLC.html>

<https://www.pjrc.com/teensy/schematic.html>

<https://www.pjrc.com/teensy/pinout.html>

# In-depth description of sensor boards

Here follows an in-depth description of each sensor board with graphics of the PCB layout, the schematics and a list of components.

## Analog accelerometer (MMA3202)

How the MMA3202 analog 2-axis acceleration sensor works is described in the previous section, and will not be described further here. On both axes the output will be at half the input voltage. Since the student rocket encoder is a 5V based system, which is also the sensor input voltage, the output will be 2.5 volts when the axis acceleration is 0g. This is called the voltage offset. For increasing acceleration in the x axis (remember that the x axis was ), the output will increase with 20 mV/g (). With increasing negative acceleration, however, the output will decrease with the same amount. For the y axis, that same factor is 40 mV/g (since this axis is , half of the range on the x axis).

So, as an example, when the acceleration on the y axis is 2g (), the output from the sensor on that axis will be

This is how it is in theory, but with this sensor, the offset of 2.5 volts is not super accurate. In practice, it can vary with volt. How can we then know what the acceleration is , one can ask? When the sensor is mounted in the rocket, on the launcher, we can very precisely know what the acceleration is (since we know the angle of the rocket, which is the angle of the launcher), *you* can calibrate the sensor using this information.

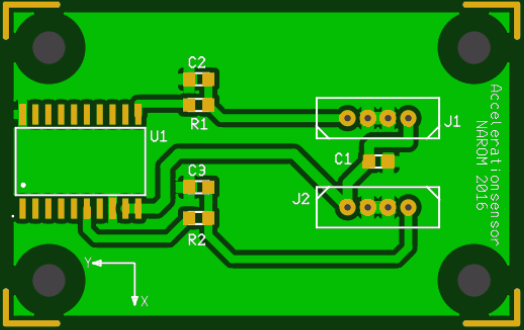
The sensor has a built in two-pole Bessel filter of 400 Hz knee frequency. And per the datasheet to reduce noise from the Bessel filter (which is a switched capacitor filter), the sensor board has a simple first-order low-pass RC-filter consisting of the resistor/capacitor pairs (R1/C2 and R2/C3). The so-called knee frequency where the amplitude has dropped to half power (called the -3dB point) of such a filter can be calculated using the equation

Datasheet

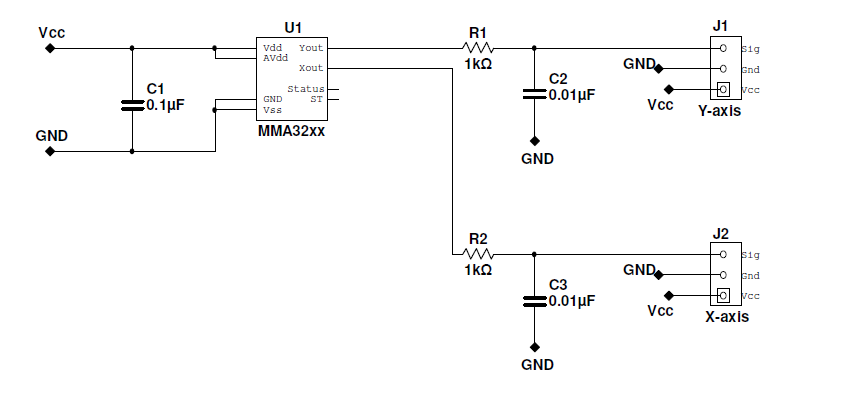
The datasheet of this sensor can be found here:

<https://www.nxp.com/docs/en/data-sheet/MMA3202KEG.pdf>

PCB, schematics and list of components

|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| R1, R2 | Resistor 1 kΩ | 2 |
| C1 | Capacitor 0.1 µF | 1 |
| C2, C3 | Capacitor 0.01 µF | 2 |
| U1 | Accelerometer MMA3202 | 1 |
| J1, J2 | Micronnector-4x1 (Datamate) | 2 |



## Magnetometer (KMZ51)

The KMZ51 one-axis very sensitive magnetic field sensor, employing the magneto resistive effect of thin-film perm alloy. It is used on the student rocket to measure the spin. We do not need to know what the magnetic field actually is, only how the magnetic field varies as the rocket spins around its own axis, so we only need to calculate the output voltage from the sensor. Please make sure that the sensor is oriented such that the measuring axis is oriented perpendicular to the axis of travel.

The sensor board has an sensor voltage amplifier using an instrument amplifier (INA122), where you as a user need to set the gain of the amplifier according to the local magnetic field strength. This amplification make sure that we use the full dynamic range of the encoder ADC (we want the output to span the full range from 0 to 5 volts). By trial and error, adjust the amplifier gain variable resistor RG while watching the output at an oscilloscope (make sure to use a time span of a few seconds on the oscilloscope) and rotating the sensor.

The sensor is known to “hang” if it has been subjected to a strong, external field. If you find that the output varies close to nothing while adjusting the gain of the amplifier, you might need to reset the sensor. The sensor needs a short pulse of about 1 A at the “Reset” gates, and NAROM has a capacitor-resistor-device that we use for this purpose. Ask your instructor for this, charge the device up to about 15 volts using a bench power supply (note the polarity of the device) and apply this to the connectors marked “Reset” (which are also polarized).

Datasheets

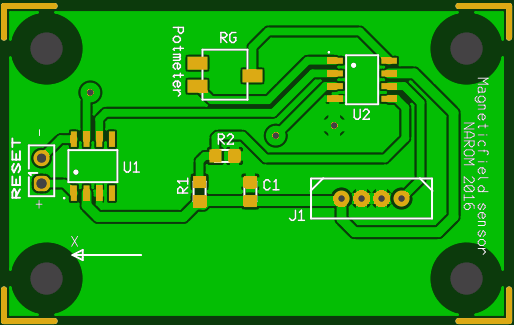
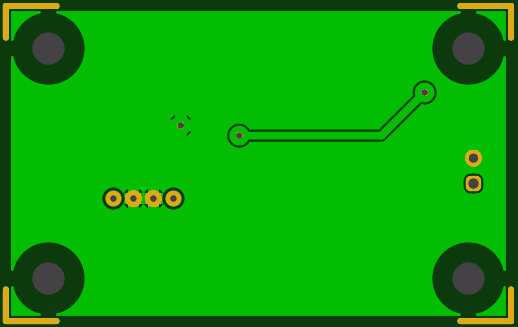
The datasheet of the KMZ51 sensor can be found here:

<https://media.digikey.com/pdf/Data%20Sheets/NXP%20PDFs/kmz51_3.pdf>

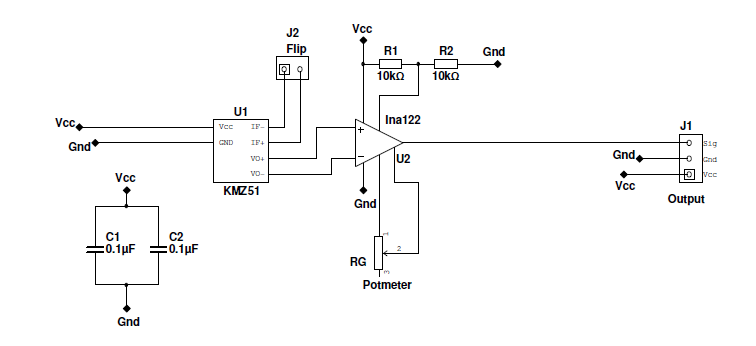
The datasheet of the INA122 instrument amplifier is found here:

<http://www.ti.com/lit/ds/symlink/ina122.pdf>

PCB, schematics and list of components

|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| R1, R2 | Resistor 10 kΩ | 2 |
| RG | Variable Resistance 0 – 2 kΩ | 1 |
| C1, C2 | Capacitor 100 nF | 2 |
| J1 | Micronnector-4x1 (Datamate) | 1 |
| U1 | Magnetometer KMZ51 | 1 |
| U2 | Instr.amp INA122UA | 1 |



## Pressure (MPX5100AP)

The chosen analog absolute pressure sensor is the MPX5100A from Freescale, now NXP (a differential variant exist in the same series). The sensor has a measurement range from 15 to 115 kPa, and is pre calibrated and temperature compensated. The pressure element is a piezoresistive transducer, and is very easy to design and construct with few external components. Note that the pressure sensor has a given orientation, so make sure to note which way you solder it on the sensor PCB. The output voltage is proportional to the external pressure.

The datasheet described how the sensor works, but in short the sensor has an internal reference pressure, and when the external pressure changes, a movable film changes the internal pressure slightly which again presses on the internal measurement element. See Figure 4 in the datasheet.

As also stated in the datasheet, the transfer function for the output voltage to absolute pressure is

where is the supply voltage which is in our case, in units of kPa. The maximum error term is (see the datasheet for more details on this), for a temperature between 0 and 85 degrees Celsius and a supply voltage of , given as

Using pressure data to calculate the altitude is very valuable. When we know the ground pressure and ground temperature, we can find the altitude from all pressure measurements using the equation

where is the temperature gradient in the troposphere, is the atmospheric specific gas constant, is the gravitational acceleration at the ground. This equation give the altitude in meters above the launch site. The ground temperature in Kelvin you need to get as accurate as possible (this is said on the intercom during the countdown and is precisely measured in the launch area) and the ground pressure (units does not matter, since this is a ratio between ground and actual pressure) is as measured from the sensor. Using the data from the sensor as (and not the most accurate pressure measurement outside of the rocket body) cancels out any error in the calibration factor on the sensor.

To see where this equation comes from, see this site:

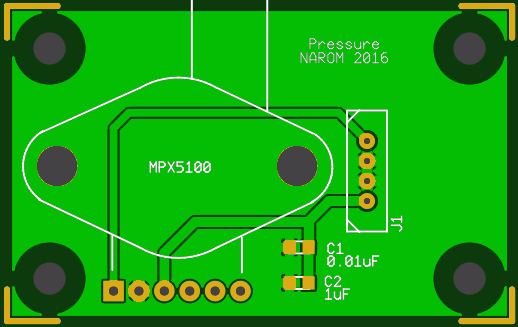
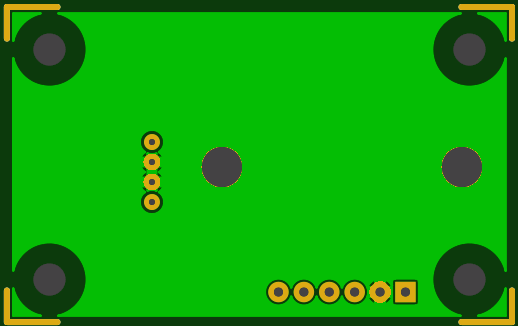
<https://www.narom.no/undervisningsressurser/the-cansat-book/the-primary-mission/using-the-sensors/altitude-calculations/>

Datasheet

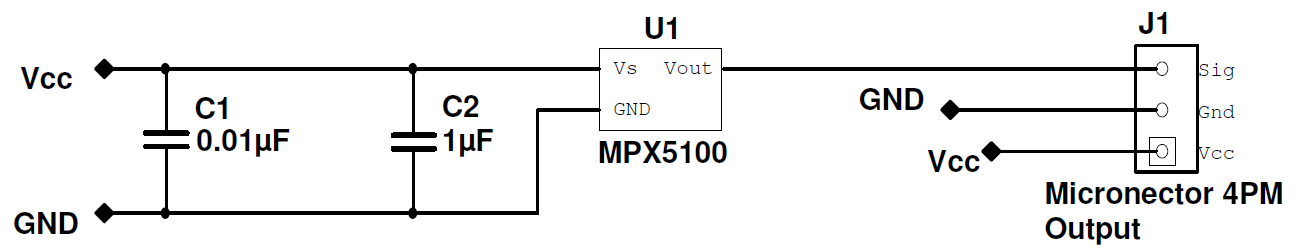
The datasheet can be found here:

<https://www.nxp.com/docs/en/data-sheet/MPX5100.pdf>

PCB, schematics and list of components

|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| C1 | Capacitor 10 nF | 1 |
| C2 | Capacitor 1 µF | 1 |
| U1 | Pressursensor, MPX5100 | 1 |
| J1 | Micronnector, 4x1 (Datamate) | 1 |



## NTC Temperature array

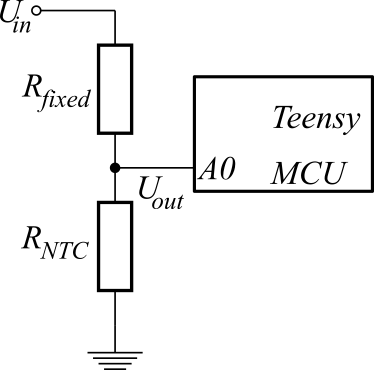
Measuring how the air drag affects the temperature on the rocket is really interesting. On this sensor board, we have a microcontroller board that measures 10 or 12 temperature sensors that is mounted/glued (by you) along the nose cone on the inside of the outer structure. When the atmosphere heats up the nose cone outer shell, the heat will spread to the inside and is what we measure by the sensors. Since we have multiple readings along the nose cone we can measure the absolute temperature, but also the temperature gradient. The nose cone has a very smooth curve with changing angles along the travelling direction, which will affects how much the skin is heated.

Contrary to what is often believed, it is not the friction from the air acting on the rocket that is heating the surface of the rocket, but an adiabatic-like process where the rocket is increasing the pressure in the air close to its body, which heats the air itself which again heat the rocket body.

So, we know that we want to measure 12 temperature sensors. We need to find a sensor that will fit our need. We want a sensor that is low-cost, has a simple interface since we want to limit the number of wires and most important: has a low mass. The reason to have a small sensor with a low mass, is generally that temperature sensors with a lower mass reacts quicker to temperature changes since it is less mass to heat up/cool down.

A sensor type that fit all of these criteria is the negative temperature coefficient sensors, or NTC for short. They are simple resistors with a resistance that is dependent on the temperature. For the NTC sensors, they have a decreasing resistance with increasing temperature, but PTC, or positive temperature coefficient, also increases. PTC would also fit our needs, but NTC is so commonly used in all types of electronics, and made in so huge numbers that they are more widespread on the market. Both would fit our needs equally good.

## Using the NTC resistor as a sensor



If the NTC sensor is to be used as a temperature sensor, we cant just measure the resistance as a voltage output. We need to design a circuit that makes us able to convert the NTC resistance as a voltage. We use a voltage divider for this purpose:

As seen in the Figure, the circuit consist of the resistors. On of the resistors is the which is the sensor itself and the other is a resistor of fixed value named . is a fixed voltage, 3.3v in our case, and the voltage on the lower side of is called ground (0 volts). In the middle of the two resistors is a voltage depending on both of the resistor values, and it is this voltage that we measure with the Teensy microcontroller board (with the A0 channel in this case, but the name of the ADC channel is not important and will vary.

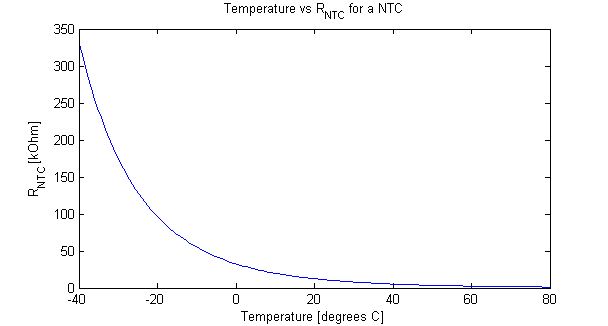
From Kirchhoff's law we know that the current that goes through must go through as long as no current go through to the microcontroller, which is a good approximation as an ADC channel is so-called high-impedance input. We write this as (the latter meaning that the current through is the same as going through both resistors at the same time. We all know that the voltage (), resistance () and current ( relationship on a resistor follows Ohm’s law, .

Ohm’s law can be rewritten as the current through the resistor , and combing this equation with with Kirchhoff's law gives us (from )

We know what the value of is (actually, you will need to find/calculate it), and we also know what is (3.3 volts, as already discussed) and is measured by the Teensy microcontroller. This only leave as an unknown, which you need to solve the equation for. *Note that this will vary during the flight.*

We still need an equation for the relationship between the resistance and temperature. For all NTC sensors this is described by the Extended Steinhart-Hart equation, usually given as

*w*here is the temperature in Kelvin, is the current NTC resistance at that temperature and ln is the natural logarithm with base is a reference value all NTC sensors has, and is the resistance value at For the sensor we have chosen, this is The constants in this equation is given in the datasheet: your sensor will have . Plotting the equation with the constants as from the datasheet, we get this resistor/temperature relationship:



Note: Often the Steinhart-Hart equation neglects the term, which often contributes little to the final result.

We have still not found the value of . What you need to do is first to find what temperature range you would expect to have. Then, calculate the resistance value of the sensor at the minimum and maximum temperature in that range. Set to the middle of those two resistance values. Resistors have only certain, specific values, so you need to take the resistor values that are closest to the theoretic value you have calculated.

The reason to find the value like you do here is to maximize the dynamic range of the ADC. Let’s take an example: we want to measure from 0 to 30 degrees Centigrade. At 0 degrees, will be about and at 30 degrees about 8 k. Since we expect to be within those values all the time, we want to measure a voltage very close to the input voltage of 3.3V when we are at 0 degrees and very close to 0 volts at 30 degrees. We want this since we then use the entire span of the measurement range we have, and get a much better resolution than choosing the wrong . For this particular example with varying from 8k to 32.5k, the middle is 20.25k. The closest possible “normal” resistor value is 20k, so this would be the best choice.

Note that in the schematics, the fixed resistance values is set to 500 ohms, but that would be a terrible choice in your case. Calculate the correct value, and use resistors you find closest to it. (Actually, 500 ohms are more rare, the closest normal value would be 470 ohms.)

## The temperature array vertical frame

So now you can work with the sensors, we need to set them in system. We know that we want to use 10 or 12 NTC sensors. First you need to find the position of all the sensors in the nose cone, then make a wire from the board to all the sensors. One sensor should have one leg connected to a common ground that all the sensors uses and the other one that goes straight to the sensor board.

On the board, the Teensy measures all these sensors and send them to the encoder. We need to make a so-called *vertical frame*, which is a frame of all these data that are stacked in the normal encoder frame. So one byte of the temperature array data is sent for each encoder frame to the ground.

The first two bytes of the vertical frame is a two byte frame sync: 160 and 161. Following that, all the twelve bit data values are sent in two bytes, so the 24 bytes following after the 161 are all the data values: each two byte data points are the 12 bits in the least significant bytes, with the first byte contains four 0’s and the four most significant bytes of the measure ADC value. A stream of data values looks like this (read this from left to right before skipping to the next line and continue left to right) with an example in parenthesis:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A7 MSB  (14) | A7 LSB  (188) | A8 MSB  (15) | A8 LSB  (93) | A9 MSB  (15) | A9 LSB  (133) | A10 MSB  (12) | A10 LSB  (120) | A11 MSB  (12) |
| A11 LSB  (67) | FS1  (**160**) | FS2  (**161**) | A0 MSB  (12) | A0 LSB  (21) | A1 MSB  (13) | A1 LSB  (197) | A2 MSB  (13) | A2 LSB  (202) |
| A3 MSB  (15) | A3 LSB  (22) | A4 MSB  (13) | A4 LSB  (157) | A5 MSB  (13) | A5 LSB  (65) | A6 MSB  (14) | A6 LSB  (88) | A7 MSB  (14) |
| A7 LSB  (157) | A8 MSB  (13) | A8 LSB  (192) | A9 MSB  (13) | A9 LSB  (20) | A10 MSB  (12) | A10 LSB  (78) | A11 MSB  (14) | A11 LSB  (2) |
| FS1  (**160**) | FS2  (**161**) | A0 MSB  (12) | A0 LSB  (190) | A1 MSB  (11) | A1 LSB  (228) | A2 MSB  (14) | A2 LSB  (171) | A3 MSB  (12) |

This will continue until we remove the power to the rocket (or splashdown for the flight). Can you see the system here? In this example, we start at some random point, A7 MSB in this case, and count two bytes for each analog channel until we have A11 LSB, which is the last byte of the frame. All new vertical frames start with the frame sync byte FS1 which is always 160. Following FS1 is the other part of the frame word, 160 which is also constant.

## Converting from ADC raw values to voltage

So how can we find the ADC raw value from this? It’s actually quite easy. Let us take the first A7 as an example. The MSB contains the most significant four bits, and the LSB contains the least significant 8 bits. So to find the raw value we just take

This is the 12-bit value returned from the ADC on the microcontroller. Since this is a 12 bit value, the maximum value we can get is 4095 (this is since we start counting at 0). So to convert this to the actual voltage between 0 and 3.3 volt, we multiply with the input voltage 3.3 and divide by 4095:

Datasheet

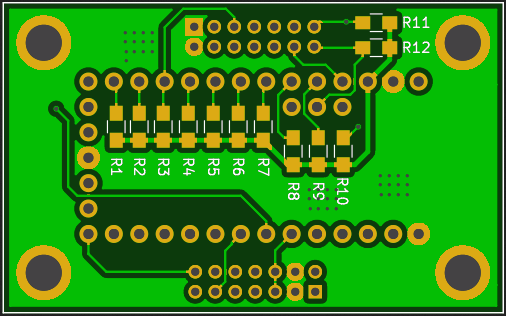
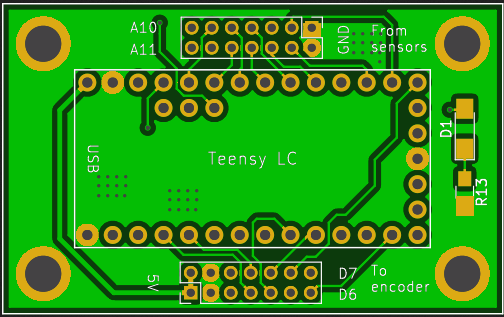
The datasheet of the NTC sensor can be found here:

<http://www.farnell.com/datasheets/2046880.pdf?_ga=2.164327679.1735290799.1539256440-468541697.1539256440>

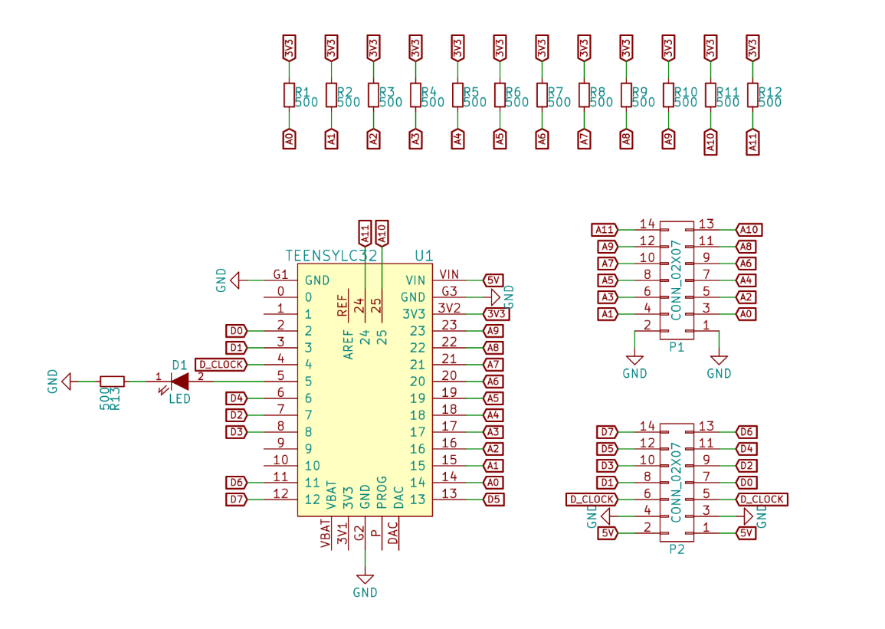
However, this datasheet is not particularly good, so if you are not looking for some very specific information, you should use instead the datasheet to a very similar NTC sensor from the same manufacturer that describes much better how this sensor works and how it should be used for everything. You can find the datasheet here:

<https://www.narom.no/wp-content/uploads/2016/11/TempSensor-NTC.pdf>

PCB, schematics and list of components

|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| R1 - R12 | Resistor (value is for you to calculate) | 12 |
| Rt1 – Rt12 | NTC Thermistor 10 kΩ | 12 |
| D1 | LED (not used) | 1 |
| R13 | Resistor 470 Ω (not used) | 1 |
| P1 – P2 | Datamate Connector 2x7 pin male | 2 |
| U1 | Teensy LC microcontroller | 1 |



## IMU (LSM9DS1)

The onboard inertial reference unit (IMU) is based on the LSM9DS1 from STmicroelectronics (a French-Italian semiconductor company) sensor which contains a 3-axis accelerometer (measures linear “motion”/acceleration), 3-axis gyroscope (measures rotation, or actually angular speed) and 3-axis magnetometer. The IMU is inertial, meaning that is does not rely on external “things” at all to measures orientation, in principle at least, except the external gravitational and magnetic field. For instance, a GPS sensor is not inertial since it relies on GPS satellites. For a vehicle that only uses IMU to navigate uses dead reckoning.

The upside for using an IMU for measuring position and velocity of a vehicle (car, drone, plane, boat) is that it does not rely on anything external except for the gravitational and magnetic field that is always there (though magnetic field can be disturbed). The downside is that the sensors in an IMU has noise, drifts and offsets, so an IMU is not accurate over time. Combining measurements from an IMU with e.g. an GPS will often be an excellent measure to navigate and locating position.

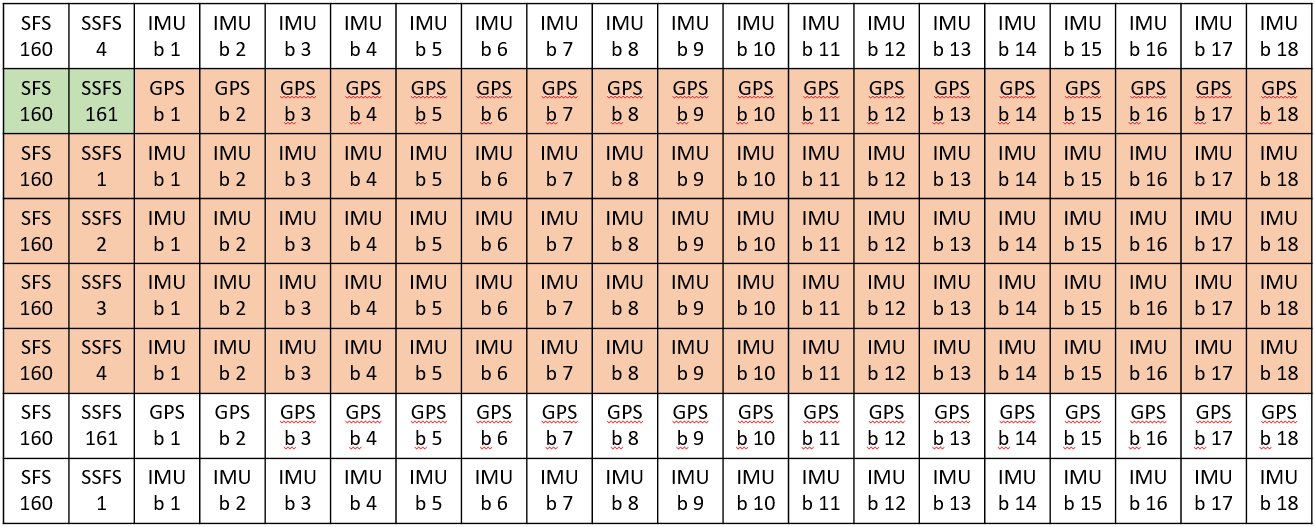
The LSM9DS1 have these measurement ranges:

* Acceleration: , , and .
* Gyro: , and
* Magnetometer: , , Converting from Gauss to the SI unit of Tesla is easy: The Earths magnetic field is about close to the poles and around at the equator.

To convert the data send from the rocket to the ground by hand is somewhat more tedious than for the temperature array sensor board, so we use a ready made MATLAB script for this purpose.

## The IMU and GPS vertical frame

For the IMU sensor board, it collects data from the IMU sensor, but also from the GPS sensor board. The reason for this is that the GPS sensor do not use that much data and is perfect to send along the IMU data. Note also that the NAROM encoder only has two digital interfaces, and one of these is already used by the temperature array.

We use the same approach as for the temperature array. Each vertical frame start with a two-byte frame sync with 160 and 161 decimal number. However, we know want to combine two different sensors. The GPS sensor is a 20 Hz sensor. It turns out that it fit quite well to have four subframes of the IMU sensor for each GPS data subframe. Each subframe in the vertical frame start with a frame sync value of 160, but only the other subframe sync words (the second of the two framesync bytes) is counting up from 1 to 4.

Each subframe consist of 20 bytes: two bytes of frame sync and 18 bytes of data. In the image above, one entire vertical frame of the IMU/GPS is denoted in colors (green and orange) and each horizontal row is one subframe. The row starting with green frame syncs is followed by only GPS data, and the other colored lines is frame sync and IMU data.

Each line (subframe) containing IMU data contains 16-bit values of accelerometer, gyro and magnetometer data in all three axis of each of the three sensors.

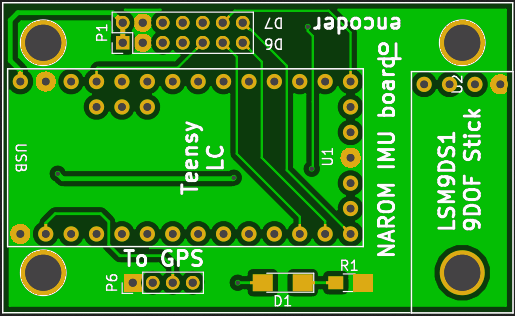
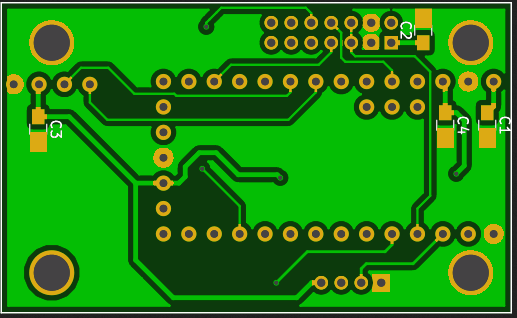
To ease the process of working with the data in post-processing, NAROM has developed a MATLAB script to read in the data from this sensor board and separate them.

Datasheet

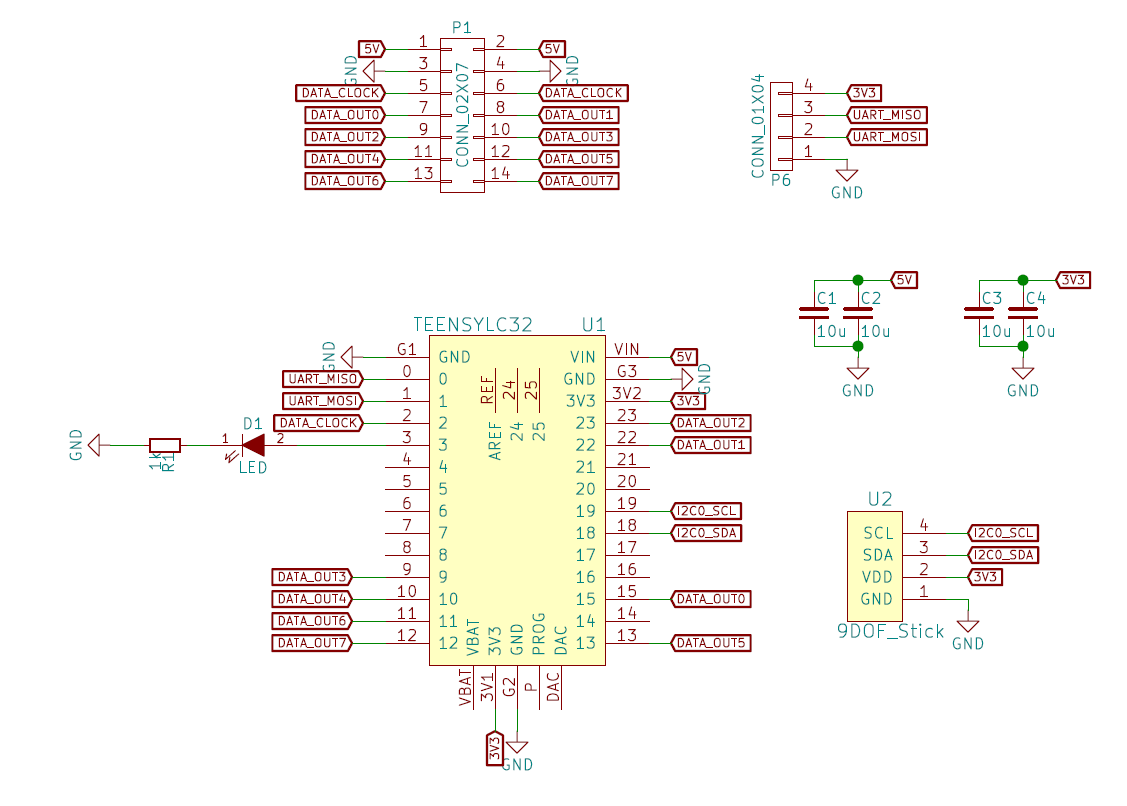
The datasheet for the LSM9DS1 sensor can be found here:

<https://www.st.com/resource/en/datasheet/DM00103319.pdf>

PCB, schematics and list of components

|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| ~~R1~~ | ~~Resistor 1 kΩ~~ (not used) | ~~1~~ |
| ~~D1~~ | ~~Light diode~~ (not used) | ~~1~~ |
| U1 | Teensy LC | 1 |
| U2 | IMU LSM9DS1 breakout board | 1 |
| C1, C3 | Capacitor 10 µF | 2 |
| C2, C4 | Capacitor 100 nF | 2 |
| P1 | Connector Datamate 2x07 | 1 |
| P6 | Connector Datamate 4x1 | 1 |



## GPS sensor (Skytraq Venus 638)

The Skytaq Venus 638 GPS sensor is few, affordable GPS ICs on the market with an update rate higher than 10 Hz. Actually, it has a output rate of 20 Hz, which is quite a lot! Many commercial GPS sensors on the market incorporates so-called COCOM limits. The COCOM limits where a legal restriction for any GPS sensor to stop output tracking data if the GPS had a speed above 515 m/s or had an altitude higher than 18 km. This restriction was enforced to make it harder for using on weapon systems, for instance missiles. The COCOM limits is not a legal restriction anymore, but it is still commonly seen in most commercial devices. However, some manufacturer only stop outputting tracking data of the device have an speed of more than 515 m/s at an altitude above 18 km. If not both of these are valid, it will still continue tracking. Luckily for us, this is true for the Venus 638, since our rocket will for sure reach a speed above 515 m/s (but be well below 18 km). This is the same sensor used on the student weather balloon PTU sonde, and here we will reach above 18 km, but not exceed a velocity of above 100 m/s.

Note that there is a very similar chip, the Venus 838, which has 50 Hz output, but there is not breakout board for it to make mounting easier on a sensor board, and hence we cannot use it.

When the GPS sensor has power, it will immediately search for a GPS lock and start output data. The default output is NMEA sentences in ASCII text format with 1 Hz data rate on its UART port. We want 20 Hz, so we have to set some registers on the GPS for it to change the UART baud rate to 115 200 baud, the output data rate to 20 Hz and from NMEA sentences to a custom binary output format (which is well described in the documentation, and hence easy to work out).

After completing soldering the sensor board(s) you will use one of the NAROM devices to load a program to alter the configuration of the GPS device.

After completing the configuration, you can take the GPS out and wait for a GPS lock and check that the position (and other data fields) make sense and seem correct. The real test would be to actually check the position with a known position, and an easy way of doing this is to use a surveying benchmark. There is one very close to Andøya Space Center at the position: 69.2989167 N, 16.0392718 E next to the MAARSY radar. Go there, set up the device, record the data and analyze the data afterwards to check how much the position deviated from the true position (which is known very precisely) and how much the GPS position changed over time. Keep the device at the benchmark for at least a few minutes to collect data looking for drifts and offsets.

Datasheet

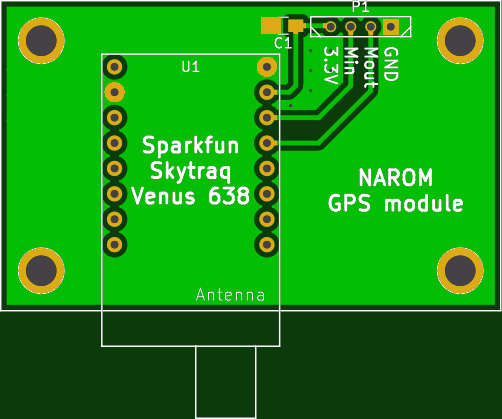
The datasheet for the Skytraq Venus 638 sensor you can find here:

<https://cdn.sparkfun.com/datasheets/Sensors/GPS/Venus638FLPx.pdf>

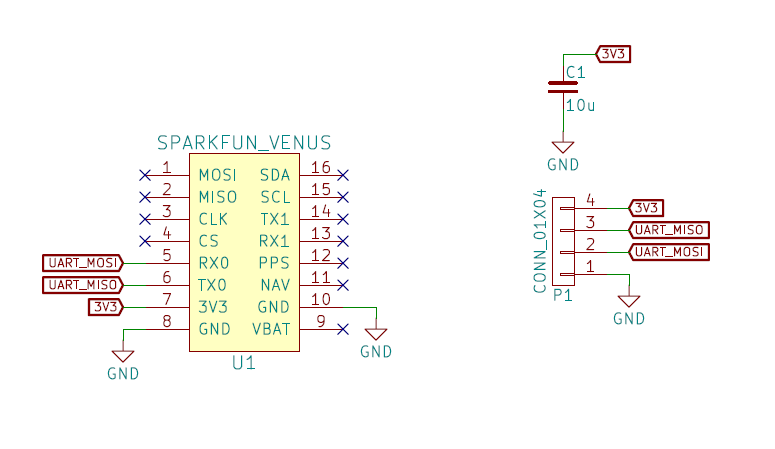
Also, the registers in the sensors is described here:

<https://www.sparkfun.com/datasheets/GPS/Modules/AN0003_v1.4.14_FlashOnly.pdf>

PCB, schematics and list of components



|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| U1 | Sparkfun Venus 638 breakout | 1 |
| C1 | Capacitor 10 µF | 1 |
| P1 | Connector Datamate 4x1 | 1 |



## PTU (weather balloon sonde)

Knowing the state of the atmosphere before and during a launch is very important, and for this we use a stratospheric weather balloon that can reach up to and above 25 km of altitude!

Building this weather balloon sonde is quite much work! The PCB has many components, you need to program it, test it well, and make a Styrofoam box to contain all the electronics and battery. The sensors that we used on the sonde is the Venus 638 GPS module, a BMP180/BMP280/BME280 module and a NTC temperature sensor. We also keep track on payload data: we measure the internal temperature (also using an NTC sensor) and the battery voltage.

Let’s discuss the most important components.

Linear battery regulator

The single cell LiPo-battery has a voltage of 4.2 volts when fully charged and the voltage will drop from this. We use a linear battery voltage named TLF1963 that is so-called low-dropout (LDO). A linear regulator as this take a certain input voltage and drops that down to a lower output voltage (a linear device like this can not step up a voltage, as a switched mode regulator may). The input current will always be the same as the output current, and the “rest” is dissipated as heat. The power which is dissipated in the regulator will then be . A linear regulator will always need to drop at least a tiny amount of power to work, so you can never have the same input and output voltage (as this will mean a ). For some well-known regulators, the minimum voltage drop can be quite large, 2 volts is often seen, but in an LDO as we use here, this voltage drop can be very small, and is often dependent on the current through the device. In out case, the voltage drop will be a minimum of around 50 mV. Our system will need 3.3V (which is not all true, but let’s say so for simplicity), so we need a battery with a capacity large enough for the battery to never drop below 3.35 volts. We use at 1000 mAh battery or larger, which will last at least a few hours.

Teensy LC microcontroller board

The Teensy LC is also used on the PTU sonde, but is described earlier in this document and is not discussed further here.

RFM96W 433 MHz LoRa radio

The radio that the Teensy will use to send data to the ground is a Long Range (LoRa) radio that uses a special radio protocol designed for transmitting low-power, low data rate data long distances. We expect to get at least 100 km of total distance of one of these beauties, though none of our balloons have had a longer distance of about 60 km. The radio send on the 433 MHz ISM band, which is open for everyone to use, and transmits data at 100 mW (+20 dBm) using a simple whip-antenna on the airborn side.

The LoRa radios work on long range, but that does come with a price: the data rate is very low! Expect a packet every about 3.5 seconds, which is enough for our use.

Skytraq Venus 638

This is the same GPS module as on the student rocket (see above), and will not be discussed further here. The main use of the GPS module is to give a position to the ground for tracking purposes (the ground station use a directional antenna which needs to point towards the sonde) and for measuring the wind. The balloon has a large atmospheric drag and will always drift with the wind. So when we see how the position changes during the flight (read: during different altitudes) we can see the wind strength and direction. Wind is of course essential to know about when launching sounding rockets, and is the main criteria when deciding if/when to launch the student rocket and in which direction.

NTC sensors

On the PTU sonde we have two temperature sensors: one internal that is used to monitor the temperature on the inside of the Styrofoam box and one external that you will place below the sonde box (take it at least 20 cm below the box, to keep the box from heating it). Using NTC thermistors as sensors is described in the section “Using the NTC resistor as a sensor”, so make sure that you read and understand that section before reading further.

The internal and external sensors will measure widely different temperatures. Expect the internal sensor to measure from 0 degrees to about 40 degrees centigrade, and the external to measure the temperature on the ground to about -70 degrees centigrade (most often it will not be that cold, but it may be, especially during the summer time, when the stratosphere is at its coldest). Based on these temperature ranges you will need to decide which you will use on the PTU sonde.

The external temperature is a must to measure when using a weather balloon, since the atmospheric layers (troposphere, tropopause and stratosphere in out case) is defined by the temperature.

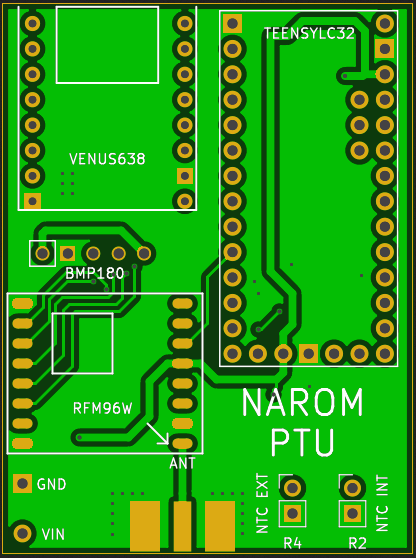
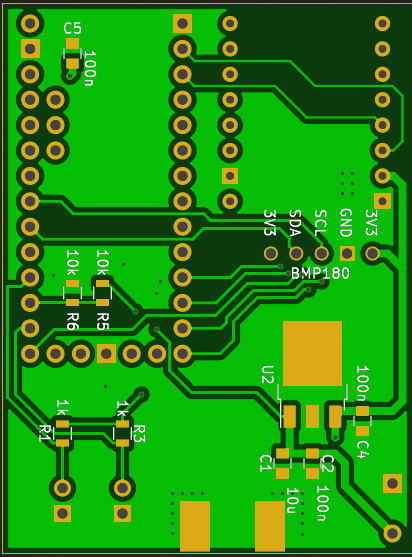
Pressure module

We need also a pressure sensor onboard, and we use either the BMP180 (second gen), BMP280 (third gen) pressure sensors or in the future BME280 pressure and humidity sensor. To measure humidity is cool, but it is not a must and is to be considered less important for our use during the student campaign.

Datasheet

PCB, schematics and list of components

*Note that the GPS, RFM96W and the Teensy is should be mounted on the side labeled “NAROM weather balloon”, and all other sensors is mounted on the other side.*

|  |  |  |
| --- | --- | --- |
| **Nomination** | **Components** | **Amount** |
| R1, R3 | Resistor: the that you need to calculate the value for for the NTC. Do **not** use 1k as on the label on the PCB. Also, R1 and R3 should not have the same values (as they are not measuring the same temperatures) | 2 |
| R5, R6 | Resistor 10 kΩ | 2 |
| R2 | Resistor NTC INT | 1 |
| R4 | Resistor NTC EXT (goes far outside of the Styrofoam box) | 1 |
| C1 | Capacitor 10 µF | 1 |
| C2, C4, C5 | Capacitor 100 nF | 3 |
| U1 | Teensy LC | 1 |
| U2 | Voltage regulator TLF1963 | 1 |
| U4 | Radio module RFM96 | 1 |
| U5 | Pressure sensor BMP180 (goes outside of the Styrofoam box) | 1 |
| U6 | Sparkfun Venus 638 GPS | 1 |

# 

# NAROM PTU template

