The CanSat Kit User Manual

The T-Minus CanSat Kit User Manual developed in cooperation with ESA's Education Office.

This version of the CanSat Kit User Manual accompanies the prototype CanSat Kit, presented at the CanSat Workshop on the 7th and 8th of December 2012 at ESTEC, Noordwijk, The Netherlands.

Contents

Introduction

The CanSat Kit User Manual was developed in cooperation with ESA's Education Office to accompany the CanSat kit. This document comprises the full description of the hardware components and the method of assembly.

Figure 1: Contents of the CanSat Kit

The CanSat kit contains (as shown in the picture from top to bottom):

- The sensor board and its components
- The T-Minus micro controller board
- A 9V battery
- The radio transmission system
- The mechanical components of the CanSat

Not included in the kit : parachute and the outer shell.

The T-Minus main controller board

The T-minus main controller board (uC) is the brain of the CanSat. The board houses the ATmega2560 micro controller and the hardware required to operate the micro controller. The extra hardware is comprised out of a power supply and a USB communication/programming interface.

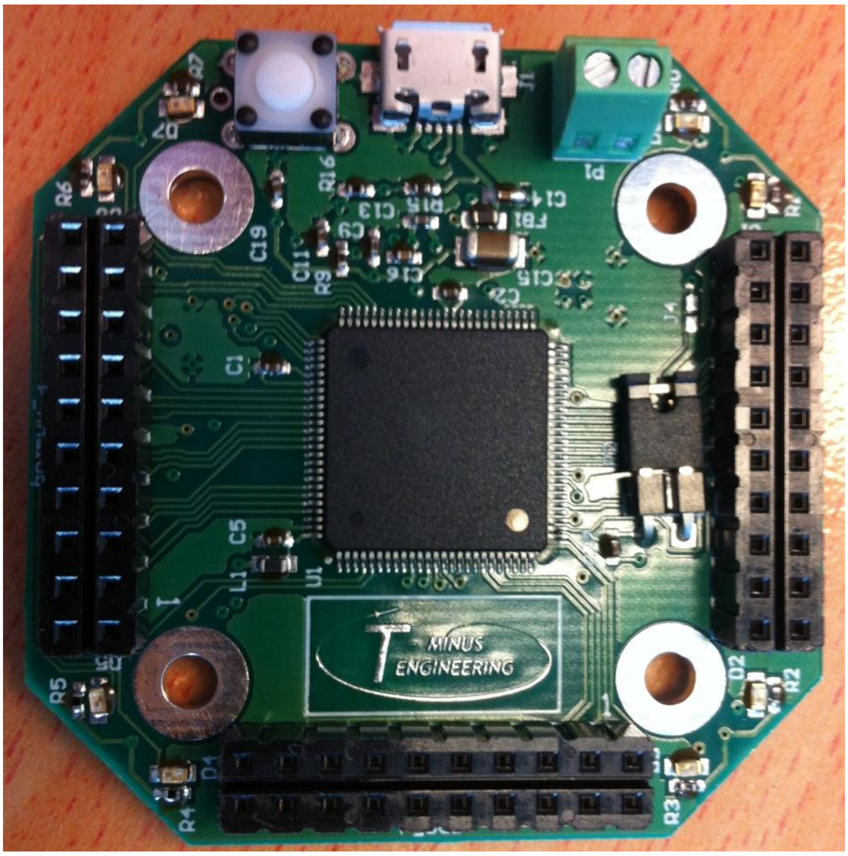


Figure 2: The T-Minus micro controller board

The power supply provides two input options for powering the device either using a USB connection or an external power supply (battery). Thus the CanSat can be powered during the testing and programming using the USB connection and during launch by using the battery. The main software environment used to programme the CanSat will be Arduino. Arduino offers a simpler method of programming the device, although it limits some of the micro controllers function.

For a more in depth control a second programming interface can be used – the JTAG. This interface allows the reach of all micro controller components and functions but will not be part of this document.

Arduino

Arduino is an open source electronics prototyping platform based on flexible ,easy to use hardware and software. Most of the information regarding programming using this environment can be found on the Arduino website (www.arduino.cc) and on dedicated forums.

The sensor board

The sensor board was designed to provide flexibility to the user. The solder able holes are each at 2.54 mm apart - the most used standard distance between the feet of electrical components. The sensor board is placed on the uC board with 3 connectors of 20 pins. These connectors have 16 data lines and 2 positive and 2 negative supply lines.

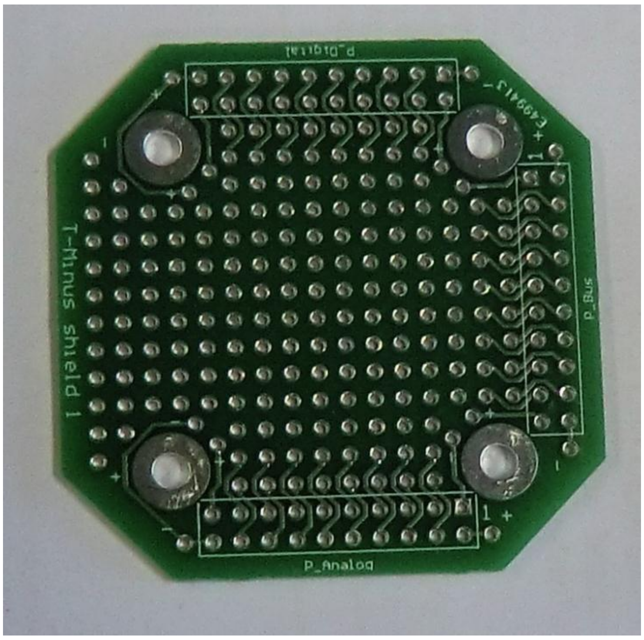


Figure 3: An empty CanSat kit sensor board

Measuring the Temperature

For the temperature measurement the kit provides two sensors. The two sensors use different methods to read the temperature, one is a thermistor and the other a an integrated circuit.

Sensor 1: The thermistor

A thermistor is a resistor where the resistance is dependent on the temperature.

The CanSat kit uses a negative temperature coefficient thermistor or NTC thermistor (the resistance of the thermistor decreases when the temperature rises)manufactured by VISHAY BC Components model NTCLE203E3103GB0. The datasheet attached as appendix shows the value of the resistor at several temperatures.

Figure 4: The thermistor (source: nl.farnell.com)

Sensor 2 : the integrated circuit

The second temperature sensor is an integrated circuit manufactured by Texas Instruments, model LM35 The datasheet of the LM35 sensor is attached as appendix. This sensor is simpler to use but give less insight in measurements.

Figure 5: The LM35 temperature sensor (source nl.farnell.com)

Pressure sensor

For the pressure measurement the kit contains a MPX4115 sensor, produced by Freescale™ Semiconsductor. This absolute pressure sensor has a pressure range of 15 to 115 kPA which is sufficient for the measurements required during the CanSat competition. More information on the sensor can be found in the datasheet which is added as appendix.

Figure 6: The MPX4115A pressure sensor (source nl.farnell.com)

Transmitter connection

The transmitter will be connected through the sensor board optimising the volume occupied by the CanSat kit components inside the shell The transmitter has a 7 pin connector to control the device and is described in more detail in the next sub chapter. Using the sensor board to place this connector provides flexibility on where to put the transmitter. If no suitable place can be found wires could be used to make the connection.

Transmitter and receiver

An APC220 communication module , manufactured by Appcon technologies comprised out of a transmitter and receiver will be used. . The APC220 is a UHF transceiver. By using two of these radios and a USB converter, a simple radio link can be obtained between the CanSat and a computer. The datasheet for the APC220 can be found in the appendix and consists out of two documents. One document describes the transceiver module and the other the manual of the complete set as it is delivered by DFRobot (www.DFRobot.com). The program RF-magic is used to configure the radios and a terminal is used for receiving the data.

Figure 7: The APC220 communication module (source DFRobot.com)

Mechanical components

The mechanical components in the kit can be used to mount the PCB to fit inside the soda can (the shell). In the CanSat kit you will find several of M3 threaded rods and an M5 eyebolt as well as nuts and rings.

The main controller board

As mentioned before, the main controller board are the brains of your CanSat. In order to make use of this board, first you need to access it via your computer.

Connecting the main controller board and the first program

The main controller board of the CanSat kit is connected to the computer via a micro USB cable. This connection is used for programming and testing of the CanSat. The first time the CanSat is connected drivers will be installed on your computer. The driver will create 2 USB devices of which one will act as a com port. When the driver is successfully installed the device will be running constantly. During the installing time it might reset from time to time. The program on the micro controller will blink the LEDs for a time of 1 second on and 1 second off. To start working with the board the Arduino program is required.

Installing

The newest version of the Arduino program can be downloaded from www.adruino.cc. To install unpack the zip file. The Arduino program does not have the TMinus uC board installed as one of its variants.

Installing T-Minus variant

To use the T-Minus uc board with Arduino, the programm needs to recognize the board. In order to do thistwo files are required to be installed. These are "board.txt" and "pins_arduino.h". The file "board.txt", located in "arduino-x.x-windows\arduino-x.x\hardware\arduino", tells Arduino how to communicate with all the board. Replace the file with the one provided. The file "pins_arduino.h" tells Arduino what pin is connected to what. This file is actually called the same for every Arduino variant. This means that instead of replacing the file it needs to be placed in a different folder. Required folder is "variants\TMinus".

Setting Arduino to use the T-minus uC board can be done by selecting the TMinus1 board under "Tools -> Board". The bottom right of the Arduino sketch window shows the board currently being used..

Initial program

In order to determine if the device works and that you can program it, you have to run a small program. To do this, the program shown below uses C. The Arduino program is on the micro controller when it is delivered.

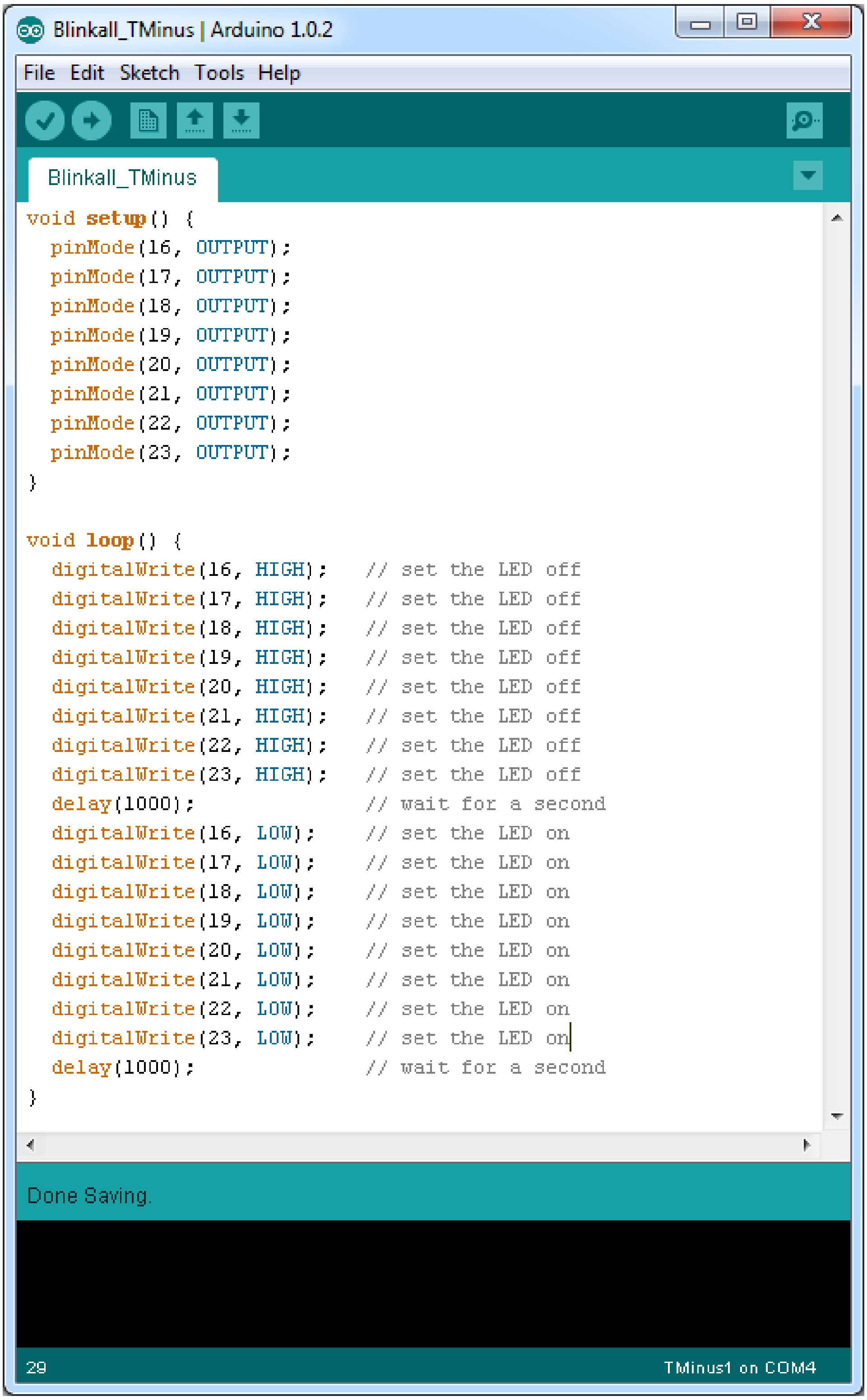


Figure 8: The Arduino program on the board when it is delivered

Step by step explanation of the program

The top of the screen indicates BlinkAll_TMinus | Arduino 1.0.2. These are the file name given to my program and the version of Arduino, in my case Arduino version 1.0.2.

The bottom right of the screen shows "TMinus1 on COM4" This indicates that I am using the TMinus1 board and that it is connected to COM4. The com port used will be different on every computer.

The program is comprised of two parts. The setup and the loop part. The Setup part of the program is run only once. This part is used to setup the controller and define initial values for variables. In the case of the original program this is used to set the micro controller outputs of the LED allowing turning the LEDs on and off.

The lines "pinMode(xx, OUTPUT);" is the call to function pinMode which is pre-defined in Arduino. The pinMode function is used to determine if a pin of the microcontroller is used as input or output. The number is a reference to the digital pin number defined in the Arduino variant being used. In the appendix the "Arduino pin to board conversion table" shows a list of the pins of the micro controller. Pins 16 to 23 are connected to the LEDs. The last part reads OUTPUT this tells the micro controller to control this pin as output, allowing two states high and low (default).

The loop part is run continuously by the micro controller from top to bottom. There are two functions used in this loop. The first is "digitalWrite(xx, HIGH/LOW);" used to determine if the assigned voltage for the a pin is 5V (high) or 0V (low). The LEDs will turn on when the pin is set as digital output low. The reason is a choice in board design. The second function is "delay(xxxx);" this makes the micro controller wait for several milliseconds dependant on the number given.

Verify and programming your initial program

The Arduino software has the ability to verify if you wrote the program in compliance with the programming rules. This verification checks if the syntax, or programming language, has been followed. The verification does not check if the program will do what you want. This is a reason to program in small steps. Verification is done by pressing the verify button in Arduino (indicated by the checkmark).

When there are no problems the program can be uploaded into in to the micro controller. To finish this process press the Upload button(indicated by the right pointing arrow

The TMinus micro controller board has a jumper. As shown in the appendix " uC board pin mapping". This jumper provides an easy way to program the board using Arduino. This also means that the micro controller is reset when USB communication between the controller and computer is initialized. For a CanSat this does not have any implications.

Communication with the main controller board

Communication is one of the most important parts of the software. While the LED's provide basic feedback that the device is working and that your program is uploaded correctly they cannot be used as an indication for the readout of the sensors. The Communication will be used to send extra information to the computer and in a later stage also to send data to the transmitter.

Basic UART communications

The basic communication of Arduino is a UART connection to the computer. The UART, or Universal Asynchronous Receiver/Transmitter, is a serial communication system with separate transmit and receive lines. On the computer this is connected to the com port. The driver installed a virtual com port for this purpose.

For sending information from the controller to the computer, both the controller and the computer need to be told what is the communication protocol. A communication protocol is how the communication is going to take place. For UART there is only 1 parameter used – the baud rate or the modulation rate – (information is sent at a certain speed measured in pulses per second) Because UART is Asynchronous this is very important or the data is read incorrectly at the receiver.

To setup the micro controller for sending communication to the computer the following function is used in the setup part of the program.

Serial.begin(9600);

Serial.begin is the function setting up the communication and 9600 is the baud rate at which the communication will run.

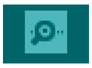
The next step in sending information is telling the program what to send. This can be done in the loop part of the program or in the setup part (while not above the begin function). There are two different functions for this:

```
Serial.print();
Serial.println();
```

The difference between the two is that the "ln" stands for, indicating that the program will send the commands for new line at the end after the printing. The data transmitted is between the brackets. To send text set it between " " like "text to be send". To send a variable place the between the brackets like (variable).

```
Serial.print("text to be send");
Serial.print(variable);
```

The last part is having the computer look at the com port. Arduino has a serial monitor build in and if the programming works the same com port is used. To open click on the serial monitor.



At the bottom right of the screen a drop down menu allows the selection of the baud rate. Setting this to the same as used in de program will show the data that is sent. Try sending text and variables with and without new lines to get a feeling of operation.

Sensor board

The next step is connecting the sensors and getting the sensor data. The sensors will be placed on the sensor board. The board used for this is the same size as the micro controller board and has connectors on the same space to place it on the micro controller board. The board is provided empty together with the connectors and components. This provides flexibility to building more components on the PCB. To start with the connectors need to be soldered on the PCB. The PCB has a top and bottom side, the top can be recognized by the markings on the PCB, the connectors should be placed on the bottom of the board, soldering on the side with the text. The other components need to go on the top of the board providing room for the components.

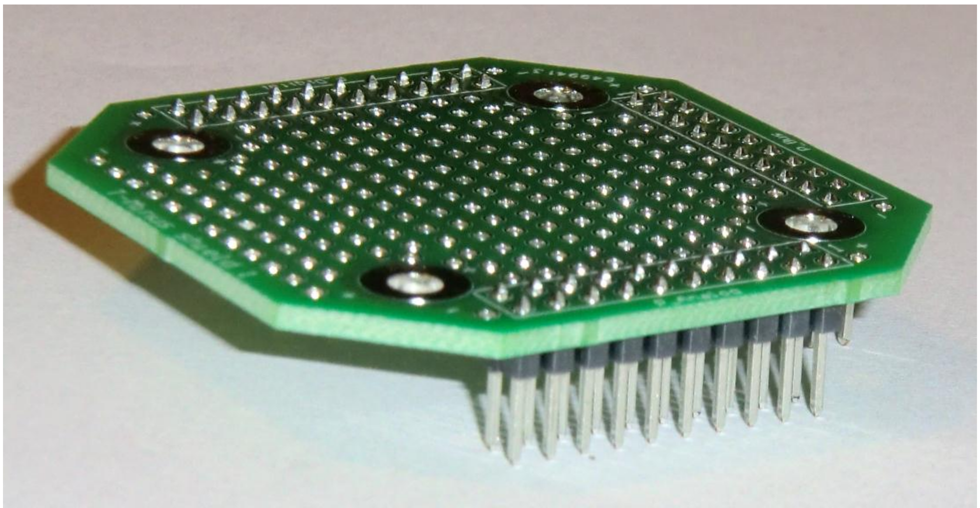


Figure 9: Sensor board PCB with placed headers

The board has many holes that are not connected while also several that are connected. The connected holes allow easy use of the connection to the micro controller board. All the pins described in appendix "uC board pin mapping" are also available on the sensor board.

Temperature sensors

There are two temperature sensors provided with the kit. Both use different methods of connecting.

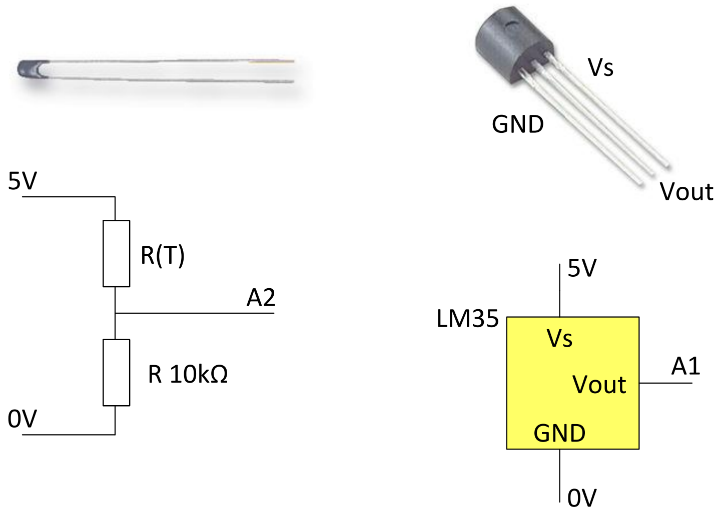


Figure 10: The temperature sensors and how to connect them

The two sensors depicted above are analogue sensors, so their output will be an analogue voltage. To measure the voltage, the sensor needs to be connected to 1 of the 12 analogue inputs of the micro controller. These inputs can be found in appendix "uC board pin mapping". 10 shows the LM35 sensor connected to the analogue pin A1 of the micro controller.

Pressure sensor

PIN NUMBERS			
1	V_{OUT} (1)	4	N/C (2)
2	GND	5	N/C (2)
3	V_S	6	N/C (2)

1. Pin 1 is noted by the notch in the lead.
2. Pins 4, 5, and 6 are internal device connections. Pin 1 is noted by the notch in the lead. Do not connect to external circuitry or ground.

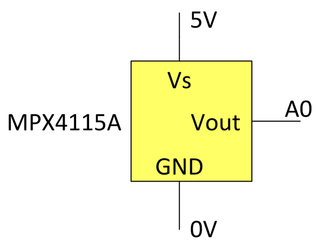


Figure 11: The pressure sensor and connection

The output of the pressure sensor is also an analogue voltage, the sensor needs to be to be connected to one of the analogue inputs of the micro controller. It is recommended to test one by one the sensors before connecting the others. This makes it easier to spot any potential problems.

Sensor readout

With a sensor connected the readout can begin. For reading an analogue value the "analogread" function is used. This functions tells the controller to read the analogue input value on one of the analogue pins an stores the value in a variable.

Variable = analogRead(Ao);

Ao is the analogue input used and Variable is the variable used to store the data. The variable needs to be defined.. There are several possible options to define a variable, the most common type used is the integer, this is a 16 bit digital value. To define a variable as integer "int" is used.

int Variable; // definition of Variable as integer

It is good practise to use self-explanatory names for variables. Simply calling them a, b, c, etc. means you will have a very hard time debugging your software.

Using the Transmitter

The transmitter for CanSat works the same way as the UART communication via the USB cable. The biggest difference is that the Transmitter needs to be connected to the micro controller. Connect 1 of the APC220 with the micro controller. Pins that need to be connected are RXD, TXD, AUX, GND, VCC. The RX of the APC220 needs to be connected to the TXD of the micro controller and the other way around. To put the APC in transmitting mode the AUX needs to be set to VCC, this can be done by connecting it to VCC or by using an GIO pin of the micro controller to switch between transmit and receive mode.

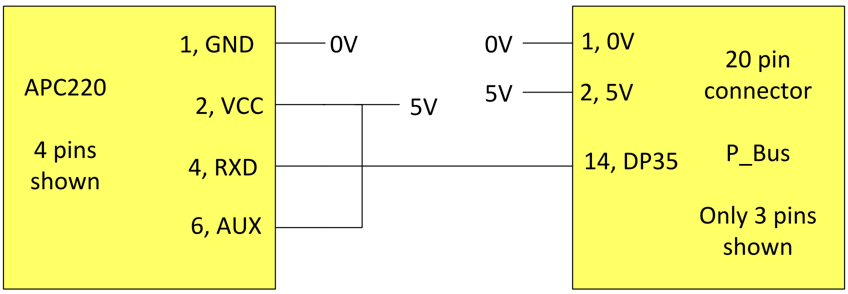


Figure 2: Schematic connection of the transmitter

In Figure the transmitter is connected to the RXD1 pin of the micro controller this is referred to as Digital Pin 35 (DP35). The AUX pin is connected to the 5V together with the VCC. This means it is always in transmit mode. The transmitter default settings use 9600 as baud rate for the UART connection. Later on it will be indicated how to change this. Using the transmitter works the same way as the connection to the computer with one difference the UART port to be used.

```
Serial1.begin(9600);
Serial1.print( );
Serial1.println( );
```

The micro controller used has 4 serial UART ports. The port used is defined by the function, Serial for UART0, Serial1 for UART1 up to Serial3 for UART3.

Receiving the signal

To receive the transmitted signal the second APC220 needs to be connected to the computer with the USB TTL delivered in the APC220 kit. When connecting the USB TTL it will need to install the drivers. These can be found on the web at:

http://www.silabs.com/products/mcu/Pages/USBtoUARTBridgeVCPDrivers.aspx

Or search for "silabs cp210x usb driver" in Google.

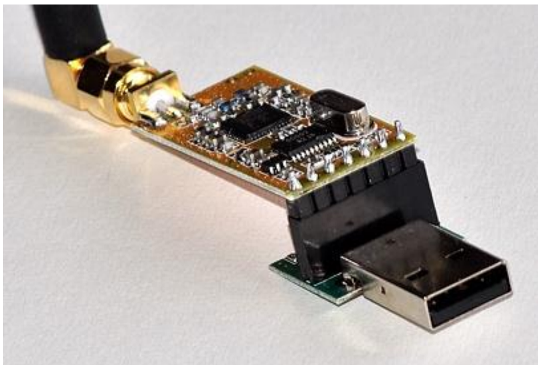


Figure 3: Picture of the APC as it fits in the USB port (source wiki.openpilot.org)

With the driver installed an extra com port has been created similar to when the micro controller was connected only with a different number. Using Arduino to read this with the serial monitor can be achieved by setting the com port to the one of the USB TTL. Before making changes to the transmitter try to experiment a bit with the software.

Changing the transmitter configuration

To change the transmitter attributes like frequency and data rates the program RF magic is used. This program can be found at several places on the internet. One place is dfrobot.com search for the apc220 and it will provide the RF-magic program as a document.

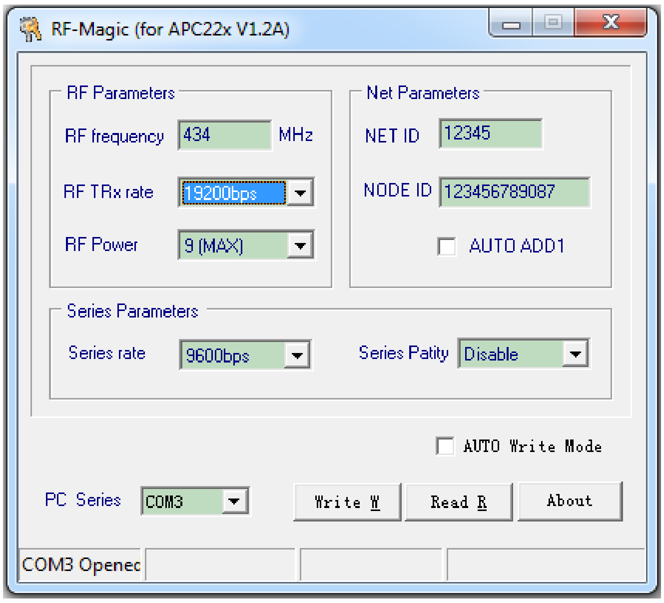


Figure 74: The RF-Magic program

The most important aspect of the transmitter to set is the frequency. The reason for this is that all CanSat teams will get a frequency allocated to them. In the program also the baud rate can be changed for both the communication through the air and the communication to the device it is connected with. Make sure to set the baud rate for the air link is the same on both sides. The Series rate does not need to be the same on both sides. They do need to be the same as what they are connected with, either the computer or the micro controller.

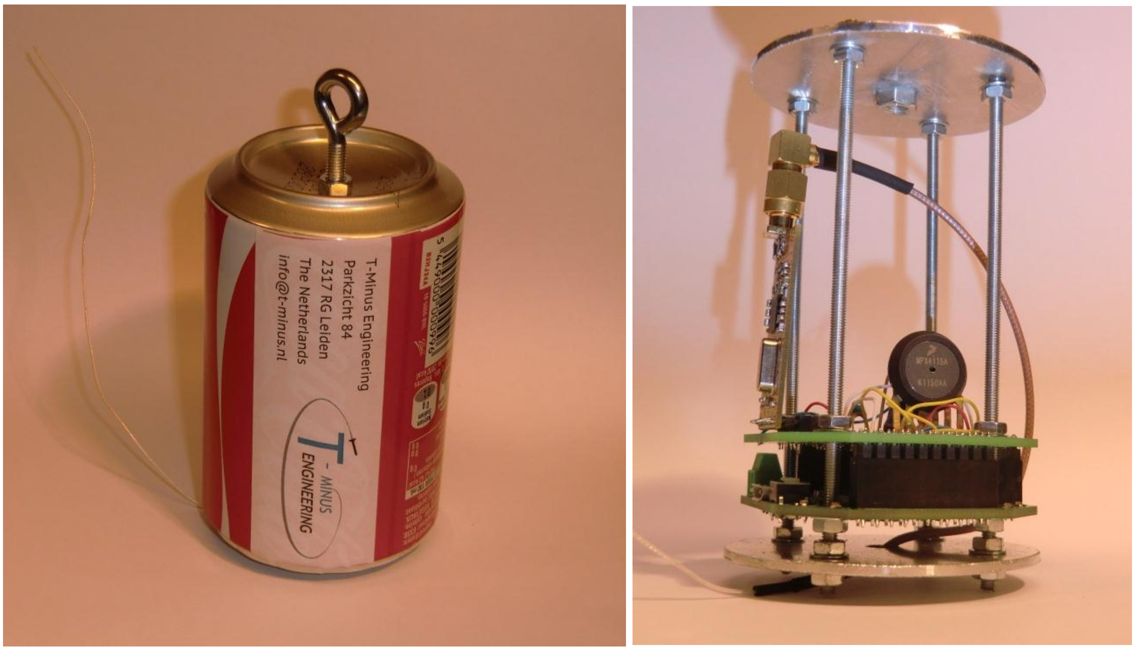


Figure 85 The CanSat with and without cover

Troubleshooting

This part will be added and expended during the CanSat competition.

Appendix A: T-Minus uC board

Specification

The board is based on an ATmega2560 micro controller. With many internal functions to view these download the complete ATmega2560 manual from the atmel website (www.atmel.com).

The power supply has two inputs a 5V USB supply. This supply is limited to 500mA and requires the FTDI drivers to operate, The FT2232 chip on the board controls the USB power this is required for proper USB operation. The drawback is that a USB power plug will not turn the device on. The second power input is the screw terminal allowing a battery or external supply to be connected. This input has a minimum of 5.5V and a maximum of 15V. The current consumption on this connector is dependent on the supplied voltage. The on-board DC-DC converter turns the supply in to the 5V used by the components of the board.

Arduino pin to board conversion table

Pin 32 to 35 can be used to generate external interrupts (see part 2). The pin numbers should correspond to interrupts into to int3. For some unexplained reason (bug), the software assigns int3 to int5 to the pins. The table shows what to use and what it will be when the problem is solved.

Arduino digital pin numbers	pin on port	connected to	function	Port	Bit of port	timer	Interrupt (IN)
0	PH0	Digital	digital I/O	PH	0		
1	PH1	Digital	digital I/O	PH	1		
2	PH2	Digital	digital I/O	PH	2		
3	PH3	Digital	digital I/O/ PWM	PH	3	OCR4A	
4	PH4	Digital	digital I/O/ PWM	PH	4	OCR4B	
5	PH5	Digital	digital I/O/ PWM	PH	5	OCR4C	
6	PH6	Digital	digital I/O/ PWM	PH	6	OCR2B	
7	PH7	Digital	digital I/O	PH	7	T4	
8	PL0	Digital	digital I/O	PL	0		
9	PL1	Digital	digital I/O	PL	1		
10	PL2	Digital	digital I/O	PL	2	T5	
11	PL3	Digital	digital I/O/ PWM	PL	3	OCR5A	
12	PL4	Digital	digital I/O/ PWM	PL	4	OCR5B	
13	PL5	Digital	digital I/O/ PWM	PL	5	OCR5C	
14	PL6	Digital	digital I/O	PL	6		
15	PL7	Digital	digital I/O	PL	7		
16	PA0	Onboard LED		PA	0		
17	PA1	Onboard LED		PA	1		
18	PA2	Onboard LED		PA	2		
19	PA3	Onboard LED		PA	3		
20	PA4	Onboard LED		PA	4		

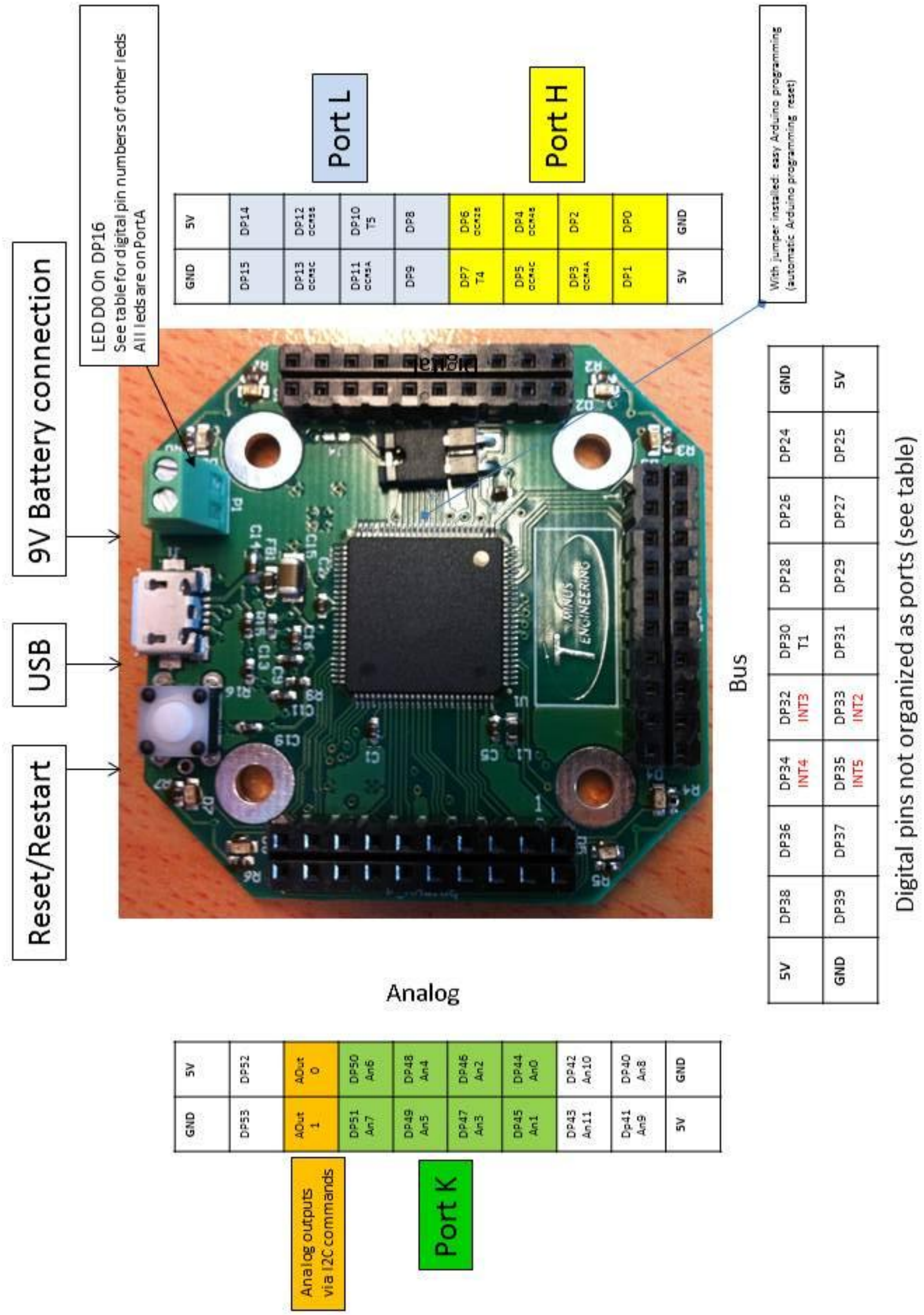
21	PA5	Onboard LED		PA	5		
22	PA6	Onboard LED		PA	6		
23	PA7	Onboard LED		PA	7		
24	PB3	BUS	MISO	PB	3		
25	PB2	BUS	MOSI	PB	2		
26	PB1	BUS	SCK	PB	1		
27	PB0	BUS	/SS	PB	0		
28	PD7	BUS	T0	PD	7	T0	
29	PB7	BUS	OC0A	PB	7	OCR0A OCR1C	
30	PD6	BUS	T1	PD	6	T1	
31	PB5	BUS	PWM	PB	5	OCR1A	
32	PD1	BUS	SDA	PD	1		1(use3)
33	PD0	BUS	SCL	PD	0		0(use2)
34	PD2	BUS	RXD1	PD	2		2(use4)
35	PD3	BUS	TXD2	PD	3		3(use5)
36	PD5	BUS	XCK1	PD	5		
37	PJ0	BUS	RXD3	PJ	0		
38	PJ1	BUS	TXD3	PJ	1		
39	PJ2	BUS	XCK3	PJ	2		
40	PF0	Analog	ADC0 (A8)	PF	0		
41	PF1	Analog	ADC1 (A9)	PF	1		
42	PF2	Analog	ADC2 (A10)	PF	2		
43	PF3	Analog	ADC3 (A11)	PF	3		
44	PK0	Analog	ADC8 (A0)	PK	0		
45	PK1	Analog	ADC9 (A1)	PK	1		
46	PK2	Analog	ADC10 (A2)	PK	2		
47	PK3	Analog	ADC11 (A3)	PK	3		
48	PK4	Analog	ADC12 (A4)	PK	4		
49	PK5	Analog	ADC13 (A5)	PK	5		
50	PK6	Analog	ADC14 (A6)	PK	6		
51	PK7	Analog	ADC15 (A7)	PK	7		
52	PE2	Analog	AIN0	PE	2		
53	PE3	Analog	PWM/AIN1	PE	3	OCR3A	
54	PE0	USB	TXD0	PE	0		
55	PE1	USB	RXD0	PE	1		

Using the analog inputs and the Analog to Digital converter

Although we do not measure analog signal values in the lab, you may want to use the `analogRead(pin)` function in Arduino, as described in http://arduino.cc/en/Reference/analogRead with the board later. Then you should use the symbolic name for the pin (A0,...A11) as shown in the table above to refer to the input pin. For example to read the analog value on pin40, the line below should be used:

```
sensorValue = analogRead(A8);
```

uC board pin mapping



Arduino install summary

Download and install Arduino from www.arduino.cc

Replace the file Board.txt with provided.
Located at “arduino-x.x-windows\arduino-x.x\hardware\arduino”
Or open the existing board.txt and add the next lines to the file.
“
##

TMinus1.name=TMinus1

TMinus1.upload.protocol=stk500v2
TMinus1.upload.maximum_size=258048
TMinus1.upload.speed=57600

TMinus1.bootloader.low_fuses=0xFF
TMinus1.bootloader.high_fuses=0x98
TMinus1.bootloader.extended_fuses=0xFF
TMinus1.bootloader.path=stk500v2
TMinus1.bootloader.file=stk500boot_v2_mega2560.hex
TMinus1.bootloader.unlock_bits=0x3F
TMinus1.bootloader.lock_bits=0x0F

TMinus1.build.mcu=atmega2560
TMinus1.build.f_cpu=8000000L
TMinus1.build.core=arduino
TMinus1.build.variant=TMinus
“

Place provided “pins_arduino.h” in the folder
“arduino-x.x-windows\arduino-x.x\hardware\arduino\variants\TMinus”

Appendix B: Component datasheets

The following component datasheets are added to this document.

- APC220 datasheet
- APC220 manual
- LM35 datasheet
- MPX4115 datasheet
- Thermistor datasheet NTCLE203E3103GB0