17. November 2006

CanSat • Milestone I

Mission Analysis and Planning

Squad 7

Cano, Angel Mario	1543440
Forslund, Lars Anders	1543952
Hakenberg, Jan Philipp	1537190
Krishnamurthy, Narayanan	1543083
Wang, Hankang	1543524
Worracharoen, Pakasit	1543191

Contents

1 Introduction

The CanSat mission objectives are listed below in order of significance:

- Collect and transmit atmospheric data.
- Demonstrate command and control capability.
- Show that an inexpensive design will survive launch loads.
- Create prototype data telemetry system and signal design for future implementation and improvement.

The main objective of the initial CanSat project is to show that the design of a small, inexpensive data telemetry system is possible. The system will have both receiving and transmitting capabilities and will be incorporated with a complement of analog and digital sensors. Although the sensor data is important for verification of the telemetry objectives, achieving individual sensor precision is not a primary goal of this CanSat.

1.1 History

Several universities have completed CanSat projects in the past, each with different missions. The University of Tokyo's project, Gekka-Beijing, consisted of three CanSats that attempted to rotate a satellite, gather temperature and pressure data, and to use a camera. They have also designed a satellite to test systems that will later be used in a Cube-Sat project. Other universities such as Arizona State University and Kyushu University have designed satellites to test a tracking system and to collect temperature data. Most of these projects are used simply to give students experience in a hands-on project. They must deal with ordering equipment, designing systems, making reports, and working in a group. This will give them an idea of what to expect when they enter the work force.

1.2 Our task description

The CanSat project is about designing a device that measures temperature, pressure and global position. The device should weigh no more than five hundred grams and have a volume no bigger than five hundred millilitres. It should be able to telecommunicate with a groundstation, sending information to a computer in order for the signal data to be processed and presented in a user interface.

This project spans over fields including telecommunication, electronics and computer programming. Moreover, it will require skills in project planning and working as a group.

2 Project management

In principle there is always more than one person working on one specific task. Our team does not have a designated leader. However, figure 2 indicates the responsibilities of each squad member.

To handle the workload required for this project, the responsibilities were split between the three group members. Hankang, Pakasit and Narayanan are in charge of programming the

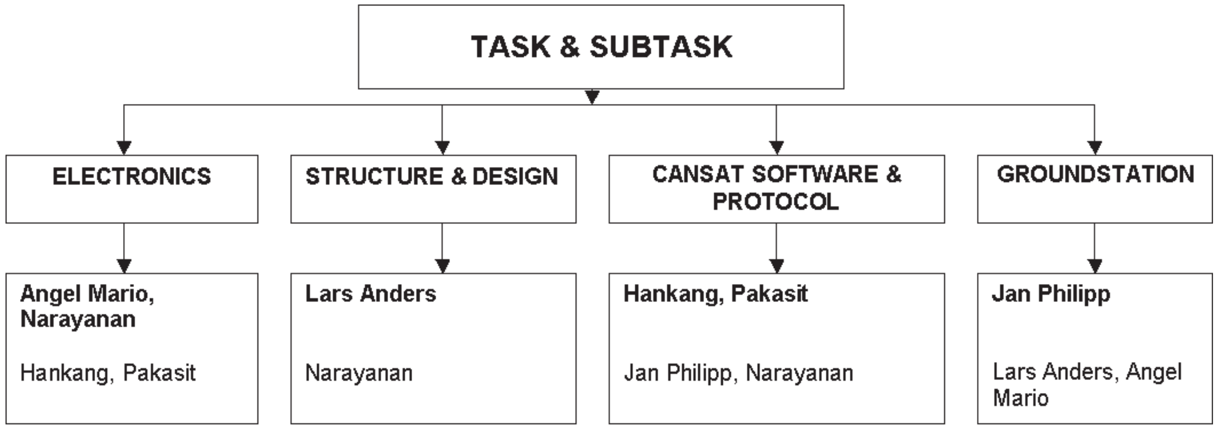

Figure 1: Overview on the subtask, listing the task manager and team members, who support the execution.

chips to work with each other and communication protocol, Angel is in charge of handling the components (such as the sensors and the circuit board), Jan is in charge of the ground station software and Anders is in charge of the structure of the satellite. The members are not restricted to working only in their area, but they specialize in one area and are responsible for making sure that area meets its requirements. The hardware integration and testing will be the responsibility of all group members. In addition, everyone contributes to the written and oral reports.

2.1 Responsibilities

Angel Mario: Electronic design of the Cansat and its subsystems is completed in its early phase, this includes: Subsystem identification and conceptual integration. Within the Cansat Electronic design we find the following subsystems: Power distribution, Data acquisition, Processing and Interfacing.

Lars Anders: I am primarily in charge of the construction of the structure of the CanSat. Since the construction is an early part of the project, I will later help develop the Java user interface. My background is engineering physics.

Jan Philipp: The core of the GroundStation software is already implemented. Screen shots are given in figure 12. As the lead programmer, I depend on Pakasit, and Hankang mainly concerning the communications protocol, timing, and testing. I am looking forward, to make the application as flexible and easy to use as possible. However, stability and recovery from communication loss are the main priorities.

Hankang: I am responsible for main program, communication module, and communication protocol improvement, system intergration, system debug, document organization.

Narayanan: I am responsible for the Project management. In the initial stages, I will be working on the Cansat Circuit design and the circuit layout. As a part of the cansat programming, I am doing the data acquisition of the pressure sensor and the Microcontroller programming.

Pakasit: I take care for the GPS data-acquisition module, temperature data-acquisition module, and especially help to debug hardware.

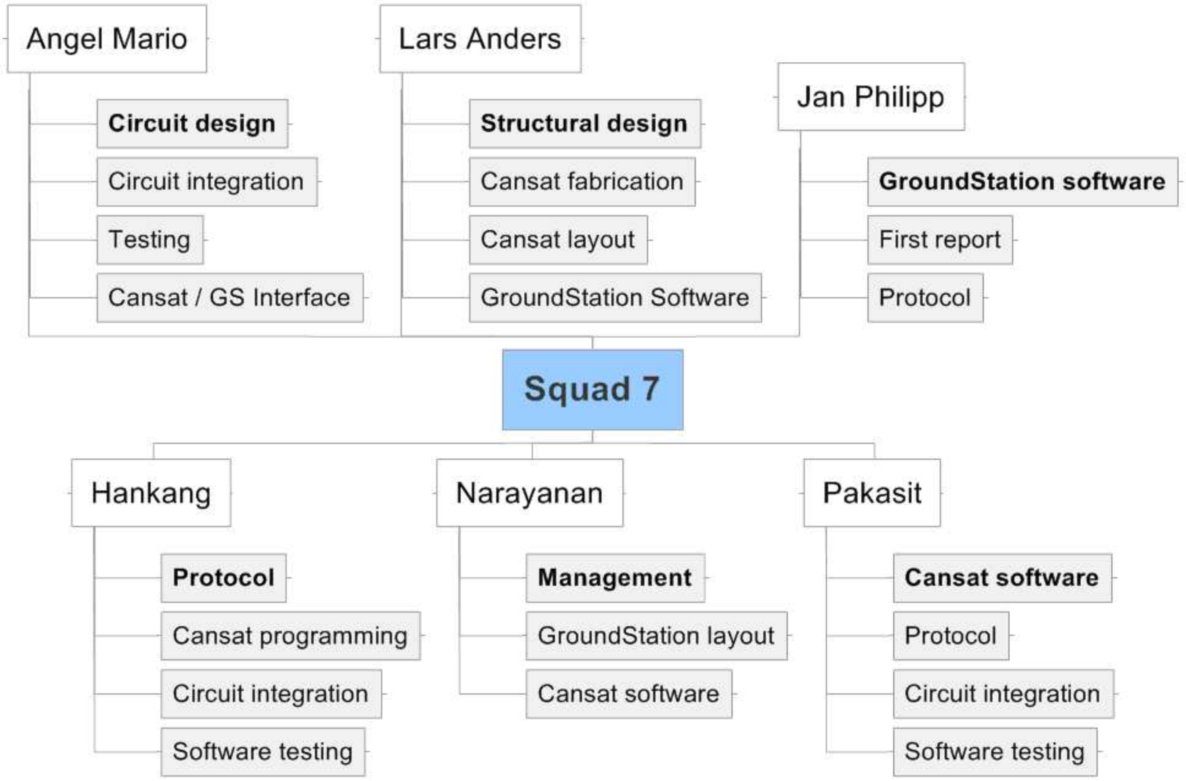

Figure 2: The team members are arranged in alphabetical order. The responsibilities of each team member appear in order of priority.

2.2 Schedule and work plan

In figures 3, and 4 below, we list tasks we have already accomplished, and task that we intend to finish in the near future. We plan to finish the cansat project in time, without making the administration angry. However, our time scedule is very tight, although we are not going for cansat tuning. That is, we are only fulfilling the absolute minimum low requirements. However, our time scedule feels very tight. If no other group is adding extra features to their cansat, we might have a chance to win the compentition.

TRAFFIC LIGHT STATUS	Color
Project on Track	
Delayed Completion	
Out of Deadline	

#	Activity	Responsiblity	Date	Status	Remarks
1	**M1: Mission Analysis and Planning**				
	Define Task	TEAM	10.10.06		Completed
	Allocation of task	TEAM	16.10.06		Completed
	Time schedule/ Work Plan	TEAM	20.10.06		
	Identify the Different Subsystems	TEAM	23.10.06		Completed
	System Architecture	TEAM	23.10.06		
	Hardware requirement/Structural Req	AND	25.10.06		Completed
	Circuit design	ANG/NAR	27.10.06		Completed
	Data Transfer Protocol(Algorithm)	HAN/PAK	27.10.06		Completed
	JAVA Programing/Front End	JAN	27.10.06		
	Final review meet of Mission 1	TEAM	30.10.06		
	Submit report		**31.10.06**		Resubmit
2	**M2:Implementation and Integration**				
	Integration of Ground Station	HAN/PAK	3.11.06		Completed
	Ground Station Programming	JAN	3.11.06		
	Testing of ground station/Hardware	JAN/PAK	10.11.06		
	Integration of CANSAT	ANG/NAR	17.11.06		
	Implement subsystems	HAN /ANG/PAK	17.11.06		
	Structure Fabrication	AND	17.11.06		
	Review Meeting-check point	TEAM	20.11.06		
	Implement test grounds for each subsystems	HAN/ANG/JAN	24.11.06		
	CANSAT Programming	PAK/NAR/JAN	1.12.06		
	Integrate various subsystems together	ANG/HAN	1.12.06		
	Final Review meet of Mission 2	TEAM	9.12.06		
	Submit report		**12.12.06**		
3	**M3:Test Evaluation**				
	Identify potential error sources/Suggest improvement		17.12.06		
	Implement test grounds for the complete system		14.01.07		
	Final Review meet of Mission 3		20.01.07		
	Submit report		**23.01.07**		
4	**M4:Final Presentation**				
	Complete project presentation		**30.01.07**		

Figure 3: High resolution time table, where we list each and every tasks with respect to their order of accomplishment.

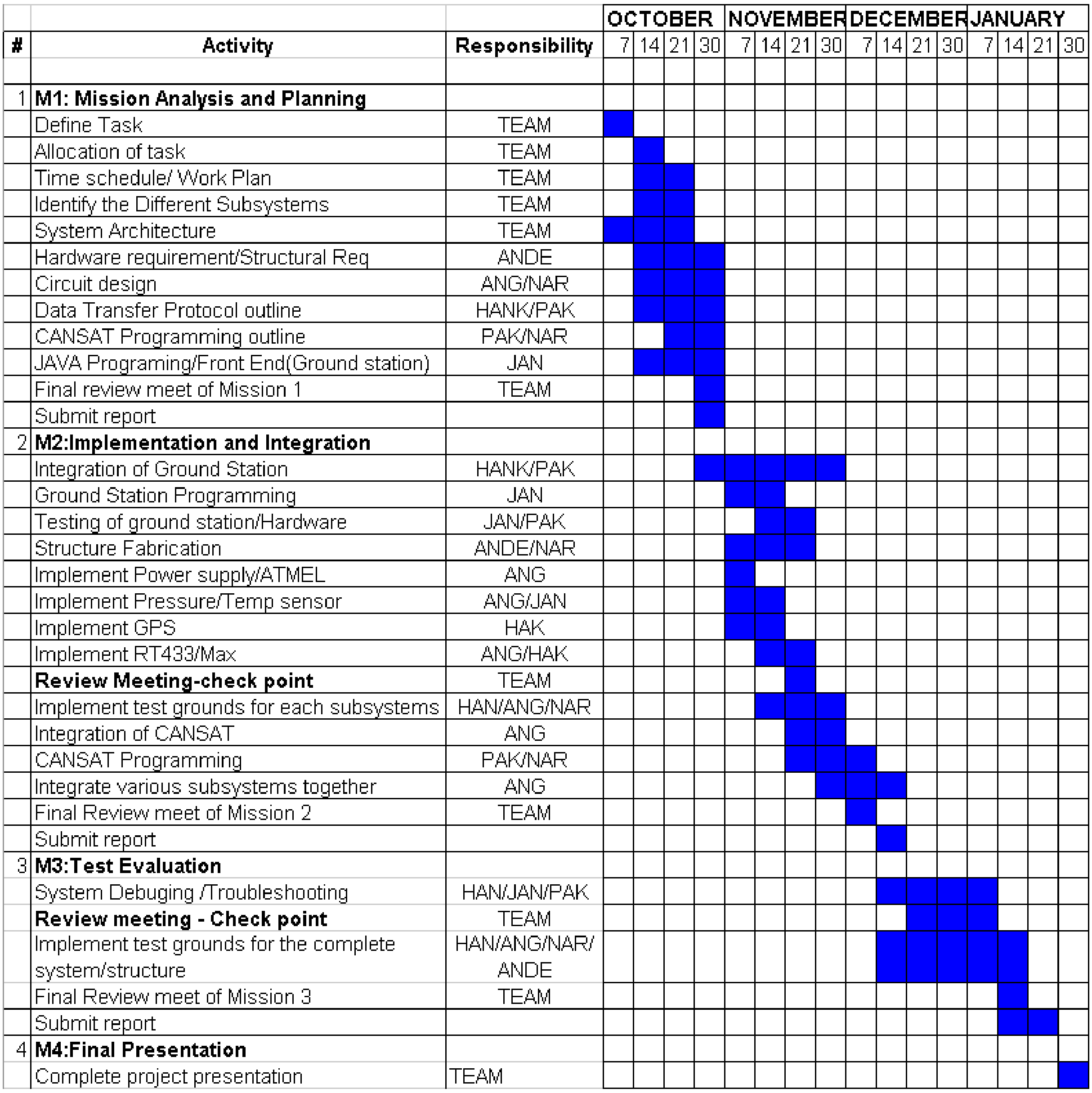

Figure 4: An alternate very detailed view on our planned time, and project management in detail.

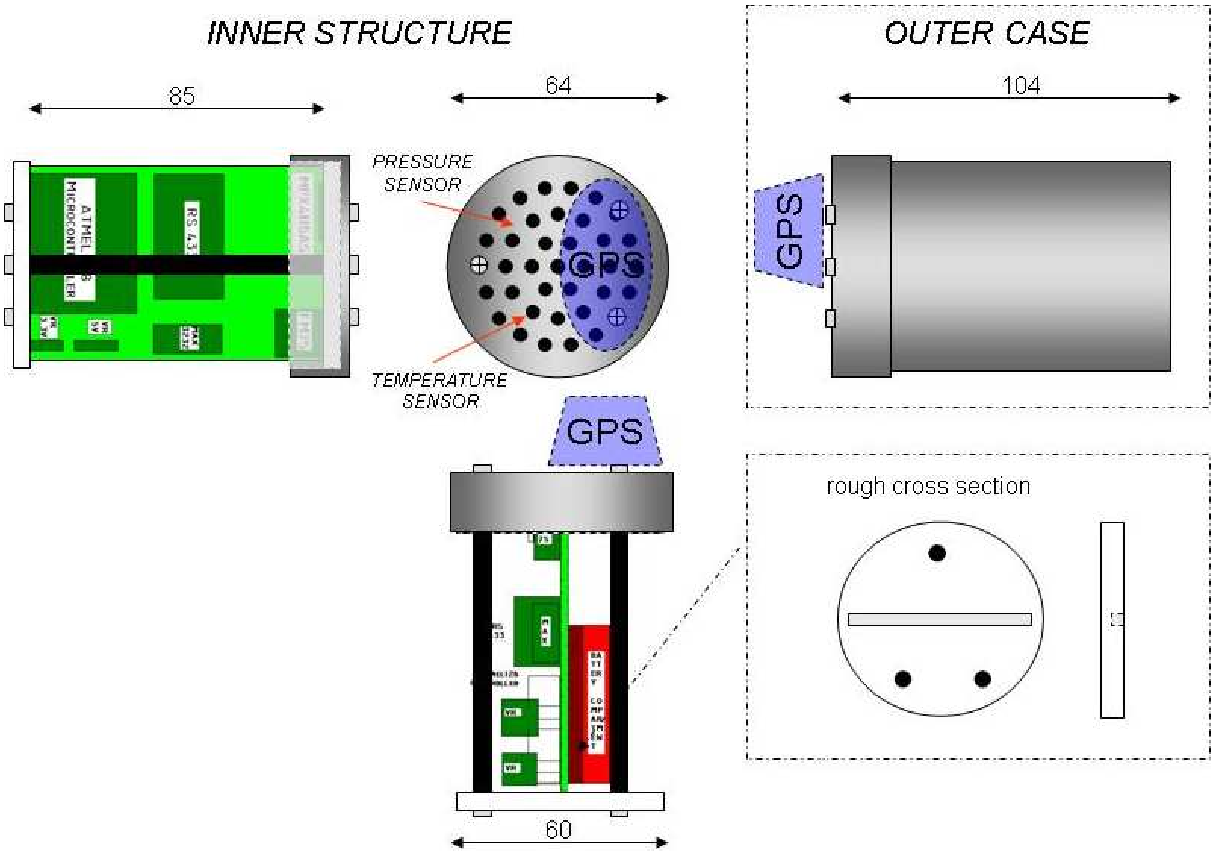

Figure 5: We visualize the assembly of the CanSat. From the aesthetic viewpoint, the jar's appealing design will give our CanSat a professional look and feel. Note, that also in the picture, the light shading indicates, that our cansat has a round, circle like cross-section. In reality, the light is reflected in the same way, because the shell is made from aluminium. This material has the advantage of not being destroyed quickly.

3 Cansat hardware

3.1 Physical construction

The CanSat device fits in a $0.5l$ can, weight less than $0.5kg$ (including batteries), and is resistant to moderate accelerations and shocks. The device acquires, filters and preprocesses data from a temperature sensor, a pressure sensor, and GPS-receiver. The data is constantly transmitted to the ground station.

The device receives and answers commands. Upon communication failure the sensor data is temporarily stored in an onboard buffer. However, the device recovers autonomously from a temporary communication loss, and re-initiates transmission of - possibly buffered - data.

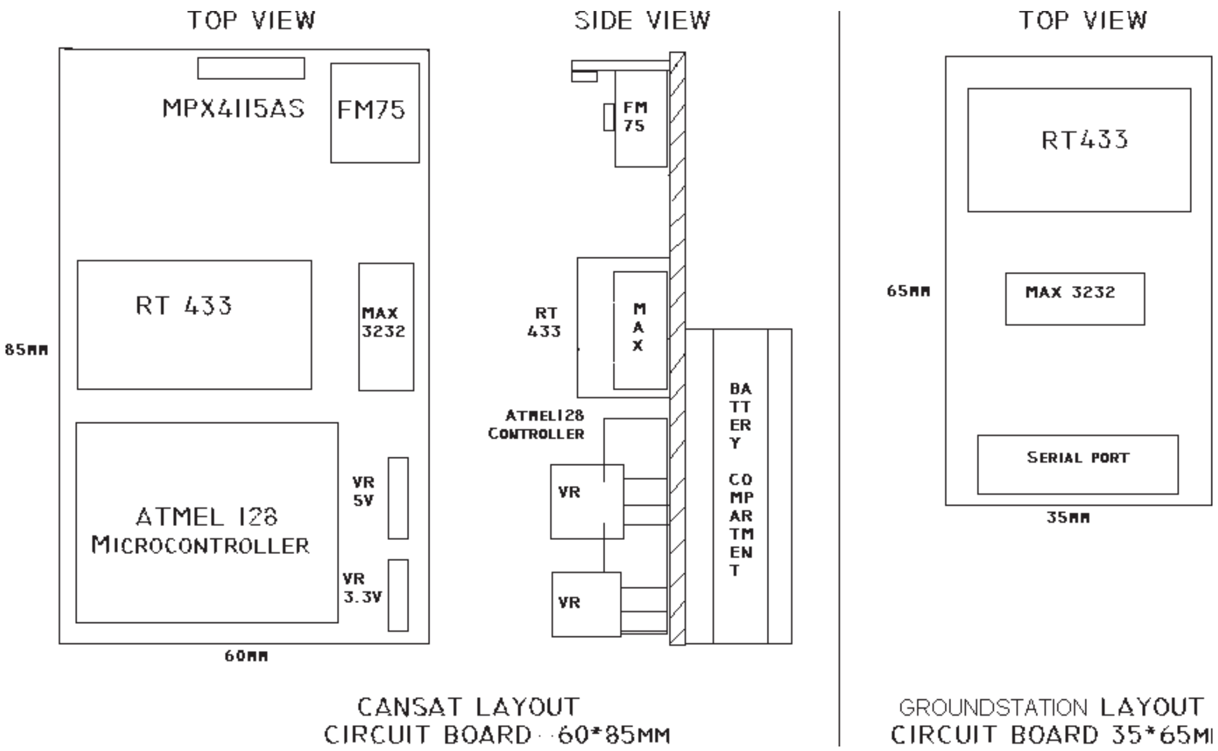

Figure 6: Even more accurate view on the circuit boards, which we will assemble. However, when we have soldered the circuits, the position of the single components on the circuit board might differ slightly from what can be seen from the pictures. Also, minor components are ommitted in the pictures.

Part	Description and features
Battery	9V supply
Crumb 128 module	microcontroller ATmega128 board, memory 128kB, supply $2.70 - 5.50V$, temperature range $0 - 70°C$
RT433F4	transceiver module, supply $2.70 - 3.30V$, temperature range $-20 - 70°C$, speed thru cable 9600, 19200, 38400 baud, frequency 433.19, 433.34, 433.50, 433.65, 433.80, 433.96, 434.11, 434.27, 434.42, 434.57 MHz, signal strength -8, -2, 4, 10 dBm
Antenna	impedance 50Ω
USB Cable	USB Cable + connector
WT12	mini-toggle switch 1xU 28VDC 3A
MAX3232CPE+	RS232E $3V - 5.5V$ DIP 16
FM75	temperature sensor I^2C interface, supply $2.70 - 5.50V$, temperature range $-40 - 125°C$, sensor precision $1°C$
MPX4115A	pressure sensor, supply $4.85 - 5.30V$, $-40 - 125°C$, pressure range $15 - 115kPa$, sensor precision $4.5kPa$
Holux GR-213	GPS receiver, altitude $< 18000m$, supply voltage $4.50 - 5.50V$, temperature range $-40°C$ to $80°C$.
TS2940CZ-5.0	Voltage Regulator $5.0V$ 1A TO220
TS2940CZ-3.3	Voltage Regulator $3.3V$ 1A TO220

We are building our CanSat using a stainless steel jar bought in a kitchenware store. This jar has many suitable properties, especially the screw-on lid perforated with 2mm holes. The volume of the jar is approximately 330ml, well under the designated maximum volume of 0.5 liter. The weight of the jar is 110 grams.

As for the construction, we plan to build the whole structure into the lid, leaving the actual jar virtually untouched. This design will ensure easy maintenance, as the circuit board can be accessed simply by screwing off the lid.

The inner structure is based on circular plastic plates that are attached to three cylindrical rods. Long indentations on the plates, that fit the contour of the circuit board, keeps the circuit board in place. The rods do not make contact with the circuit board.

As the GPS receiver is said to require "open sky" to work properly, we decided to put it on top of the lid. This will increase the total volume, but we will still be far from exceeding the maximum volume. The pressure and temperature sensors will be placed just under the holes on the lid, giving them direct contact with the outside environment. We will try to make the entire construction sealed by blocking the rest of the holes (besides the hole used for the GPS receiver cord).

The weight of the components, inner structure and jar is approximately 300 grams. This leaves 200 grams for battery, wiring etc., which we think is more than enough.

3.2 Circuit

The main theoretical background required for the design of the CanSat involves circuit theory. A firm understanding of the relationship between current and voltage and the role of capacitors, resistors, and inductors in circuit design is imperative for the completion of this type of project. The task will also require knowledge of how to convert a radio frequency signal to digital data and programming microchips to carry out those conversions. Programming microchips to handle radio signals necessitates an understanding of how binary data is stored in data registers, the different types of registers, the implementation of interrupt logic, and amateur radio protocol for transmitting data. Most of these topics are more closely related to electrical engineering than aerospace engineering and are thus outside the bounds of an aerospace undergraduate preparation. A large learning curve has been associated with these topics; given below is a brief synopsis of those topics and recommendations as to what future CanSat groups may need to research before attempting to improve on this CanSat design.

3.3 Sensor specifications

We give a detailed description of the sensors the cansat is equipped. This includes operating ranges, precision, and error estimation.

- **Temperature:** Accordint to [FM75], the precision of the temperature sensor is $\pm 1°C$ in the range of $0°C$ to $100°C$. Hovever, the operating range is $-40°C$ to $125°C$. The operating current of the sensor is less than $250\mu A$. The device only heats itself at most $0.2°C$ in still air.

- **Pressure:** According to [MPX4115A], the precision of the temperature sensor is $\pm 1.5\%$ in the range of 15kPa to 115kPa, and $0°C$ to $85°C$. The full temperature operating range is $-40°C$ to $125°C$, with loss of precision at the extrema. The error computation is displayed in the data sheet [MPX4115A].

- **Global positioning:** The Holux GPS device is a comparable large device. Its dimensions are $(64.5, 42, 17.8)$mm. The temperature operating range is $-40°C$ to $80°C$. The Holux GPS device tracks up to 20 satellites. The update sensoring rate is 1 second. Initialization time takes about 38 sec.

3.4 Power management

As source of power on the CanSat device, we will choose either a $9V$ Alkaline Battery with approximately $500mAh$, or a $9V$ NiCd Battery with approximately $250mAh$. In the following table, we summarize the power consumption of the active components of the CanSat device.

Part	Description	max mA	Operating V	Power mW
ATmega128	Microcontroller	35	5	95
RT433F4	Transceiver module	60	3,3	198
FM75 (first)	Temp. sensor - I^2C interface	11	5	55
FM75 (second)	Temp. sensor - I^2C interface	11	5	55
MPX4115A	Pressure sensor	8	5	40
Holux GR-213	GPS receiver	80	5	400
MAX3232CPE+	RS232 2xDr./Rec. 3-5, $5V$ DIP	20	3,3	66
MAX3221CPE+	RS232 2xDr./Rec. 3-5, $5V$ DIP	20	5	100
TS2940CZ-5,0	Voltage-Reg +5,0V	30	9	270
TS2940CZ-3,3	Voltage-Reg +3,3V	30	5	150
Total		305		1374

From this we compute an estimated running time of

1.639344 hours	using the Alkaline Battery
0.819672 hours	using the NiCd Battery.

4 Cansat Software

At this time, our CanSat has only one analog sensors. The difference between the types of sensors is discussed below.

An analog sensor records data continuously, while a digital sensor defines data in several individual ”steps”. The main advantage of an analog sensor is its ability to fully represent a continuous stream of information without eliminating any information.

Digital sensors, on the other hand, are less affected by unwanted noise and interference, and therefore provide a clearer signal. These types of signals are often converted from one to the other. For example, this conversion occurs every time a signal is sent over a phone line. A transmitting (encoding) modem converts digital data from a computer to analog sounds and the receiving (decoding) modem then converts the analog signal back to the original digital data for analysis. Because of these characteristics, analog signals are used to transmit data over long distances and digital data is required for computer analysis.

In the following, we discuss sensor data-acquisition modules.

4.1 GPS receiver

4.1.1 Operational characteristics

Initialization: As soon as the initial self-test is complete, the GR-213 begins the process of

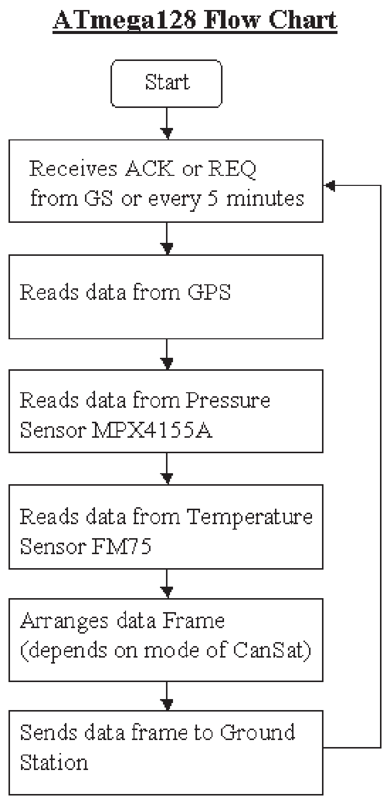

Figure 7: The illustration displays the principle flow of the ATmega128 program. Due to testing the software, we might modify this layout.

satellite acquisition and tracking automatically. Under normal circumstances, it takes approximately 42 seconds to achieve a position fix, 38 seconds if ephemeris data is known. After a position fix has been calculated, information about valid position, velocity and time is transmitted over the output channel.

The GR-213 utilizes initial data, such as last stored position, date, time and satellite orbital data, to achieve maximum acquisition performance. If significant inaccuracy exists in the initial data, or the orbital data is obsolete, it may take more time to achieve a navigation solution. The GR-213 Auto-locate feature is capable of automatically determining a navigation solution without intervention from the host system. However, acquisition performance can be improved as the host system initializes the GR-213 in the following situation:

1. Moving further than 500 kilometers.

2. Failure of data storage due to the inactive internal memory battery.

Navigation: After the acquisition process is complete, the GR-213 sends valid navigation information over output channels. These data include: Latitude/longitude/altitude, velocity, date/time, error estimates, satellite and receiver status

4.1.2 Software interface

The GR-213 interface protocol is based on the National Marine Electronics Association's NMEA 0183 ASC interface specification, which is defined in NMEA 0183.

NMEA-0183 Transmitted Messages: The default communication parameters for NMEA output are 4800 baud, 8 data bits, stop bit, and no parity.

NMEA-0183 Output Messages:

GPGGA	Global positioning system fixed data
GPGLL	Geographic position- latitude/longitude
GPGSA	GNSS DOP and active satellites
GPGSV	GNSS satellites in view
GPRMC	Recommended minimum specific GNSS data
GPVTG	Course over ground and ground speed

4.1.3 Data frame format

See the GR-213-manual-E, [JP7T] for many more details. Here, we only list the most important frames.

1. Global Positioning System Fix Data (GGA)

   `$GPGGA,161229.487,3723.2475,N,12158.3416,W,1,07,1.0,9.0,M, , , ,0000*18`

2. Geographic Position with Latitude/Longitude (GLL)

   `$GPGLL,3723.2475,N,12158.3416,W,161229.487,A*2C`

3. GNSS DOP and Active Satellites (GSA)

   `$GPGSA,A,3,07,02,26,27,09,04,15, , , , , ,1.8,1.0,1.5*33`

4. GNSS Satellites in View (GSV)

   `$GPGSV,2,1,07,07,79,048,42,02,51,062,43,26,36,256,42,27,27,138,42*71`

5. Recommended Minimum Specific GNSS Data (RMC)

   `$GPRMC,161229.487,A,3723.2475,N,12158.3416,W,0.13,309.62,120598, ,*10`

6. Course Over Ground and Ground Speed (VTG)

   `$GPVTG,309.62,T, ,M,0.13,N,0.2,K*6E`

7. ZDA-SiRF Timing Message

   `$GPZDA,181813,14,10,2003,00,00*4F`

4.1.4 Setting Syntax

Manufacturing Defaults:

Datum	WGS84.
Baud rate	4800.
Output	GGA, GSA, GSV, RMC.

Datum change syntax: `sirfprog /Fdataxx.dat -Px -Bx -Csh1`, where the suffixes are used as, -Px: x is com port, 1= COM1, 2 = COM2 -Bx: Baud rate, 4800, 9600, 19200 or 38400

For instance, if you would like to change datum to WGS84, use the command `sirfprog Fdata58.dat -P1 -B4800 -Csh1`. After changing datum, the new datum will be kept in SRAM. If no power supplied to GR-213 for more than 30 days, user must re-set datum when power on.

4.2 Temperature Sensor

4.2.1 Basic Operation

The FM75 temperature sensing circuitry continuously produces an analog voltage that is proportional to the device temperature. At regular intervals the FM75 converts the analog voltage to a two's complement digital value, which is placed into the temperature register.

The FM75 has a SMBus compatible digital serial interface which allows the user to access the data in the temperature register at any time. In addition, the serial interface gives the user easy access to all other FM75 registers to customize operation of the device.

The FM75 temperature-to-digital conversion can have 9, 10, 11, or 12-bit resolution as selected by the user, providing $0.5°C$, $0.25°C$, $0.125°C$, and $0.0625°C$ temperature resolution, respectively. At power-up the default conversion resolution is 9-bits. The conversion resolution is controlled by the R0 and R1 bits in the Configuration Register.

The thermal alarm has two modes of operation: Comparator Mode and Interrupt Mode.

1. **Comparator Mode**: The new digital temperature is compared to the value stored in the T(OS) and T(HYST) registers.

   If a fault tolerance number of consecutive temperature measurements are greater than the value stored in the T(OS) register, the O.S. output will be activated

   Once the O.S. output is active, it will remain active until the first time the measured temperature drops below the temperature stored in the T(HYST) register.

2. **Interrupt Mode**: This mode will first become active after a fault tolerance number of consecutive temperature measurements exceed the value stored in the T(OS) register.

   Once O.S. is active, it can only be cleared by a user read from any of the FM75 registers or by putting the FM75 into Shutdown Mode.

   It can only be activated again by a fault tolerance number of consecutive temperature measurements that are lower than the vale stored in T(HYST)

   Once it is activated the O.S. output can only be deactivated by a user read or shutdown.

4.2.2 Data format

1. **Command Register**

   ```
   MSB               LSB
   0   0 0 0 0 0 P1 P0
   ```

   The data in Command Register (8-bit) indicates which of the other four registers (Temperature, Configuration, T(OS), or T(HYST)) the user intends to read from or write to during and upcoming operation. The P1 and P0 bits of the Command Register determine which register is to be accessed.

2. **Temperature Register**

   ```
   MSB 14   13 12 11 ...                        1 LSB
   SB  TMSB T  T  T  T T T T 9b 10b 11b 12b 0 0 0 0
   ```

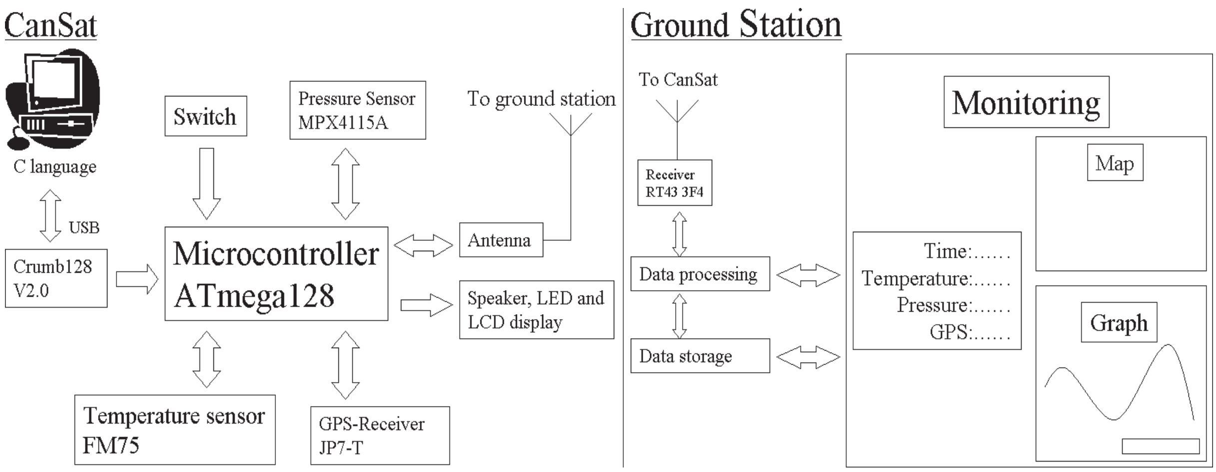

Figure 8: Schematics of the CanSat device as well as the Groundstation. Only the most relevant components are included.

SB	Two's complement sign bit
TMSB	Temperature MSB
9b	Temperature LSB for 9 - bit conversions
10b	Temperature LSB for 10 - bit conversions
11b	Temperature LSB for 11 - bit conversions
12b	Temperature LSB for 12 - bit conversions

3. **Configure Register**

```
MSB                         LSB
X1 R1 R0 F1 F0 POL CMP/INT SD
```

R1	Resolution bit 1.
R0	Resolution bit 2.
F1	Fault tolerance bit 1.
F0	Fault tolerance bit 2.
POL	O.S. output polarity. 0 = active low, 1 = active high.
CMP/INT	Thermostat mode. 0 = comparator mode, 1 = interrupt mode.
SD	Shut down. 0 = normal operation, 1 = shutdown operation mode.

4. **Over-Limit-Signal Temperature Register**, also referred to as T(OS)) / Hysteresis Temperature Register (T(HYST)

```
MSB 14   13 12 11 ...                       1 LSB
SB  TMSB T  T  T  T T T T 9b 10b 11b 12b 0 0 0 0
```

5. **Slave Address**

```
1 0 0 1 A2 A1 A0
```

4.3 Pressure sensor

The output of the pressure sensor is in a form of analog voltage. Port F in the ATmel128 serves as the analog inputs to the A/D Converter. So, we must define and distribute port F for Pressure data acquisition module.

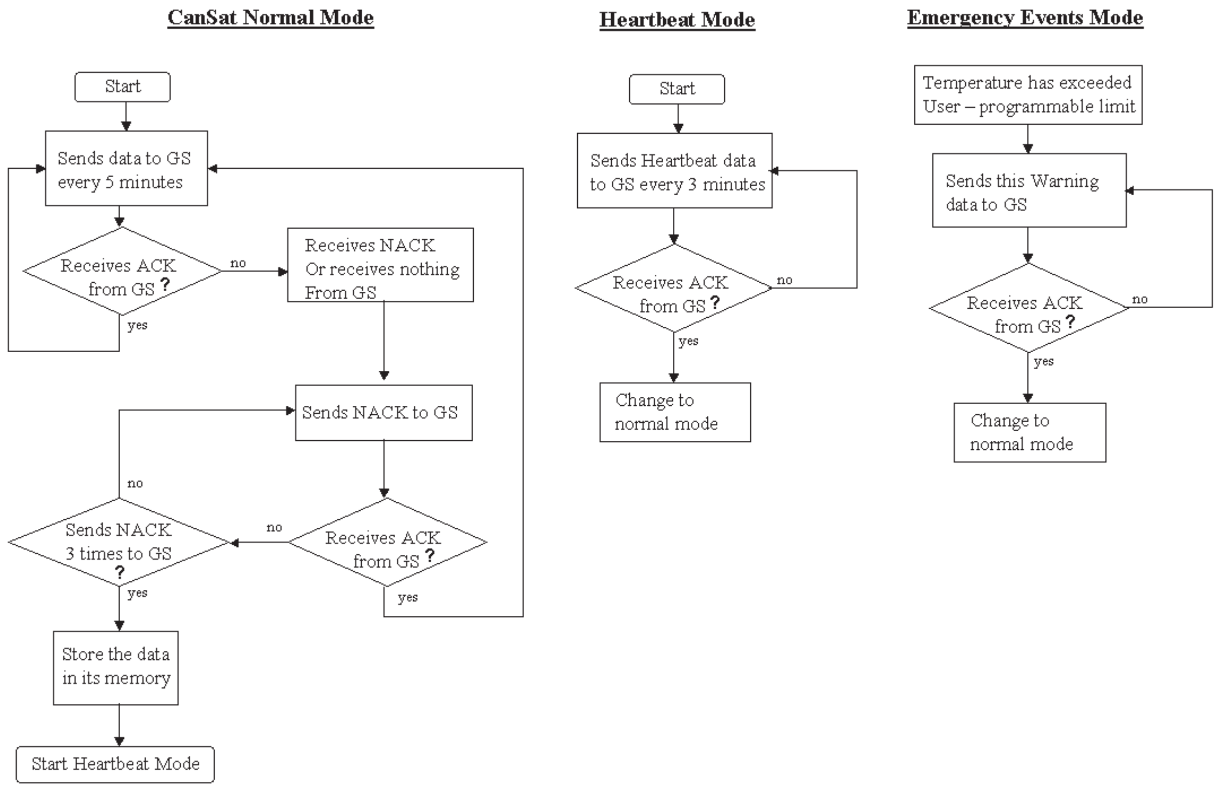

Figure 9: We display the flow charts that characterize the possible communication modes.

Nominal Transfer Value :

$$V_{out} = V_S \times (0.009 \times P - 0.095) \pm (\text{Pressure Error} \times \text{Temp. Factor} \times 0.009 \times V_S)$$

$$V_S = 5.1 \pm 0.25V\text{dc}$$

According to the hardware, the bottom layer of pressure module is to read the output from the sensor and calculate the Vout with respect to the pressure error and temperature factor. The input from temperature sensor has to be read, to match the Pressure error and the Temperature error band. The range of the temperature error factor has to be varied with respect to change in temperature. Then, the Vout has to be encoded to Hex Data. The Hex data should be wrapped and sent to the main program. The main program has to envelop the Hex data using the data read and command-write function. All the functions used in the pressure data acquisition module should be included Head Files of the main program.

4.4 Communication

4.4.1 Modes

1. *Normal Mode*: Cansat(CS) sends data to Groundstation(GS) every 5 minutes (the value can be configured differently). If GS receives data correctly and successfully, GS respond ACK signal to CS, then another repetition(loop). If GS does not receive data from CS correctly or successfully, GS responds another NACK signal to CS, telling the CS, that GS did not receive correct data stream, or receive data failed, please send again. If fails 3 times, CS would store the data in its memory (because of no network delay and disturbance, the

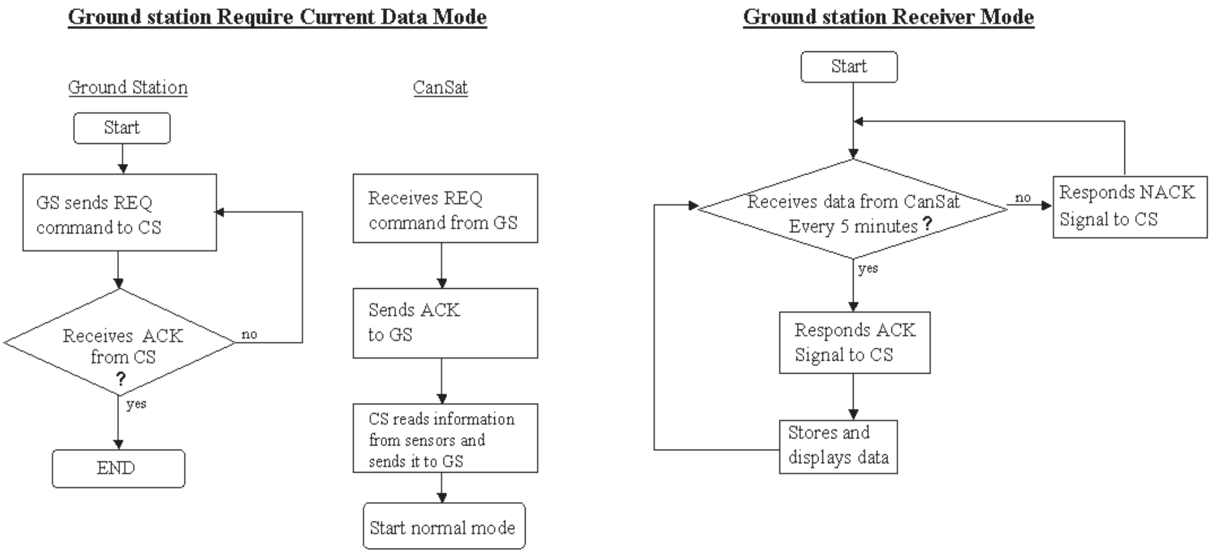

Figure 10: We display the flow charts that characterize the possible communication modes.

probability is almost 0). If CS sends data to GS, but receive no any response, repeat 3 times, then store the data, change to Heartbeat Mode*(4).

2. *Groundstation Require Current Data Mode*: GS sends Requirement(REQ) command to CS. If CS receives REQ command, echo ACK to tell GS to be ready to receive the current real data. CS reads information from sensors immediately, packages and send data to GS. Then like *Normal Mode*.

3. *Emergent Events Mode*: If the temperature in CS exceeds user-programmable limit, CS sends WARNING DATA directly to GS (No requirement to GS). Then CS waits for receiving GS ACK. Then everything like *(1).

4. *Heartbeat Mode*: Maybe CS fails to receive/send any data from/to GS. In this case, CS should send Heartbeat Data to GS in every 3 minutes (can be configured to desired values). Once it receives the Response Heartbeat ACK, it means CS and GS can communicate with each other, then send stored information, and then change to *(1).

4.4.2 Data frame format

1. Normal data frame (0x80)

```
0      6  7  8  9  10 11 12 13 14 15  <N+M+L>  16+M+N+L
CANSAT 30 30 01 80 00 01                       TAIL
```

First	Last	Content
0	5	HEAD, define as CANSAT
6	8	CANSAT ID, all in ASCII CODE
9		DATA TYPE such as 0x80
10	11	DATA SERIAL NUMBER, such as 00 01
12		GPS DATA length ($\leq 255 = $ FF)
13		PRESSURE DATA length ($\leq 255 = $ FF)
14		TEMPERATURE DATA length ($\leq 255 = $ FF)
15		VERIFY DATA byte adds (XOR) all bytes of the real data, place the summation to this byte
16	16+N-1	GPS data
16+N	16+N+M-1	Pressure data
16+N+M	16+N+M+L-1	Temperature data
16+N+M+L	16+N+M+L+3	suffix, TAIL

The VERIFY DATA byte

2. Emergent Events Data Frame (0x81)

```
CANSAT  . . .   81  . . .                     TAIL
```

3. Groundstation Data ACK frame (0x82)

```
CANSAT  . . .   82  . . .             YACK    TAIL
```

4. Emergent Events ACK Data frame (0x83)

```
CANSAT  . . .   83  . . .             YACK    TAIL
```

5. GS Received Data incorrectly Response Data frame (0x84)

```
CANSAT  . . .   84  . . .             NACK    TAIL
```

6. GS Require Current Data REQ frame (0x85)

```
CANSAT  . . .   85  . . .             DREQ    TAIL
```

7. CS Respond GS REQ frame (0x86)

```
CANSAT  . . .   86  . . .             DRES    TAIL
```

8. HEARTBEAT frame (0x87)

```
CANSAT  . . .   87  . . .             HB      TAIL
```

9. HEARTBEAT ACK from GS (0x88)

```
CANSAT  . . .   88  . . .             HBACK   TAIL
```

Possibly, the Groundstation Data ACK frame will be combined with the GS Received Data incorrectly Response Data frame in the future. Also, the values of DATA TYPE could be taken from ASCII.

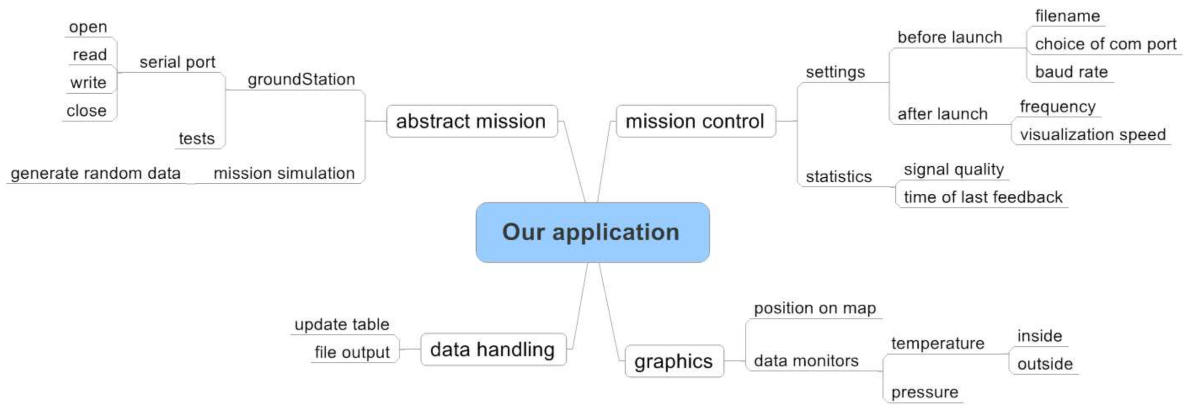

Figure 11: Structure of the GroundStation application. The `abstract` GroundStation class has various implementations to test the software.

5 Ground Station

The ground station device sends commands specified by the operator to the CanSat. The ground station receives, and processes the atmospheric data sent by CanSat. The ground station detects communication link failures, and recovers autonomously from a temporary communication loss.

5.1 Circuit

In our implementation, the ground station consists of a transciever module linked via serial port to an ordinary personal computer, which displays, and stores the received data. We develop software in Java to perform these tasks. The outline of the program is subject of the next subsection.

Part	Description and features
RT433F4	transceiver module, supply $2.70 - 3.30V$, temperature range $-20 - 70°C$, speed thru cable 9600, 19200, 38400 baud, frequency 433.19, 433.34, 433.50, 433.65, 433.80, 433.96, 434.11, 434.27, 434.42, 434.57 MHz, signal strength -8, -2, 4, 10 dBm
Antenna	impedance 50Ω
MAX3232CPE+	RS232E $3V - 5.5V$ DIP 16
Capacitors	$0.1\mu F$
BL09LR	D-Sub-connector female 9pol-RS232

5.2 Software

We develop an easy-to-use Java application to display, and store the data, which we receive from Cansat. As illustrated in figure 12, the application displays not only the atmospheric data, but also displays information on the current radio connection.

5.2.1 Key features

- The data is stored into a `.csv`-file, comma-separated value format. This means, the file is compliant to commercial applications such as MS-Excel.

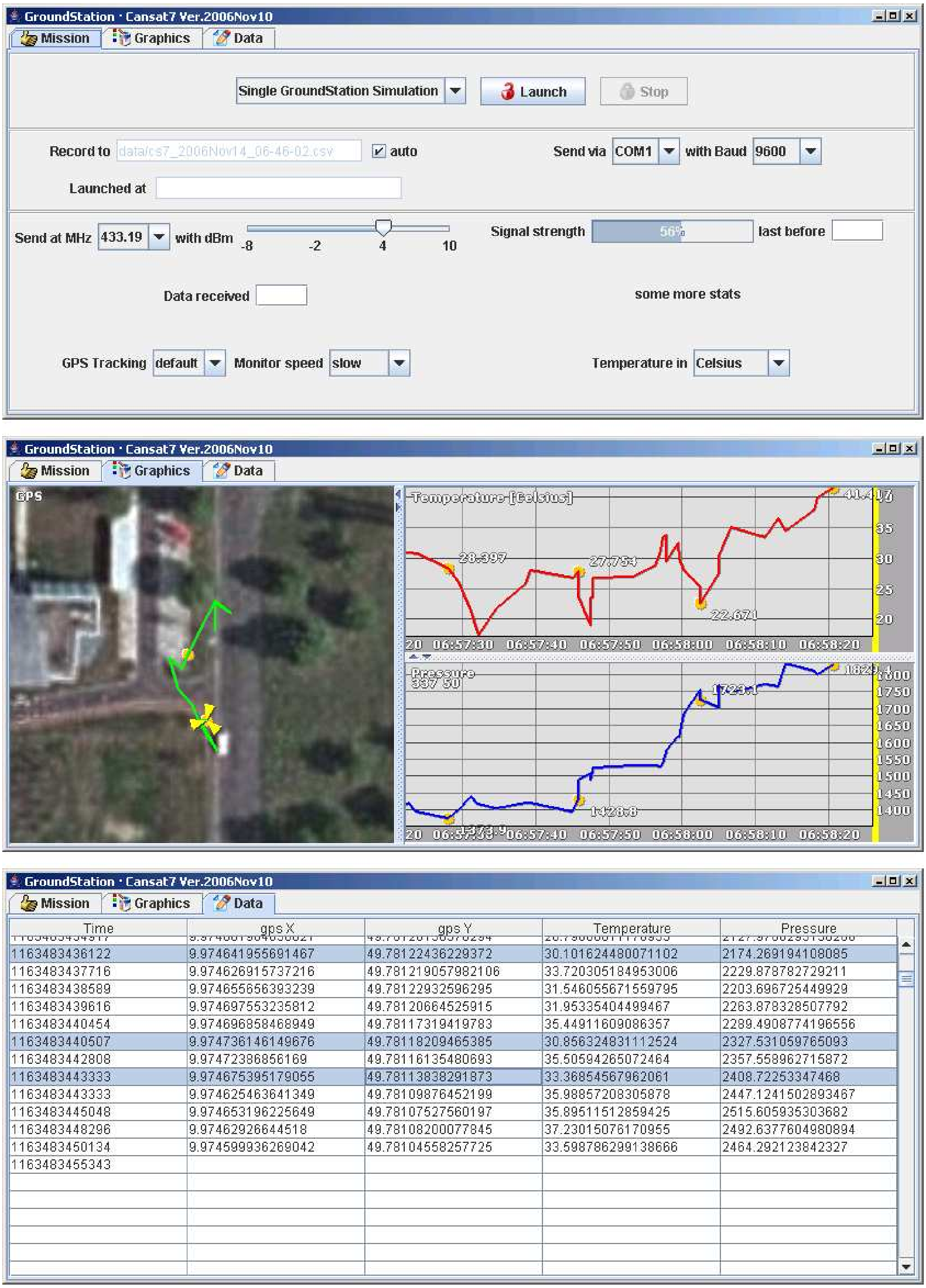

Figure 12: Graphical user interface of Groundstation with fictive physical measurements.

- Direct communication to the transceiver module of the groundstation enables us to obtain feedback about signal quality. If the quality is high, then we might send a command to the Cansat, that it lowers its signal strengh, to save pover.
- The user can select different speeds of visualization. In the graphics, the user can highlight track points to see the numerical value.
- The position of the Cansat is mapped to an prestored satellite image of the campus of the Julius-Maximilian University Würburg.

5.2.2 Communications

We will layout our code, such that the application with interactive display is encapsulated from the protocol. This is realized, by designing an abstract communication class, which is then extended to a specific protocol. Hypothetical, this allows easy integration of communications to Cansats of other groups.

Our particular protocol convention can be found in section 4.4. Except for the timing, the sending, and parsing of the data frames should turn out to be straight forward.

Upon mission launch we allocate a sufficient amount of memory, to store the data in, which we receive over the course of the mission. When there is a data frame incoming, we parse the frame according to the protocol. Our objective is to not lose any information on athomospheric data acquired by Cansat. If there is a error in the check sum received by the Groundstation, we dispose the message and ask the cansat to resend the data frame. In case of success, we send an acknoledgement frame.

For permanent storage, we flush each data frame directly to a mission log file.

5.2.3 Graphical user interface

The data acquired by the Groundstation is distributed to various modules of the Groundstation application. First, there are the graphical monitors of temperature (outside, and inside of cansat), and pressure. Then, there is the table class, which lists the exact numerical values. These systems are independent, so this policy should not be a source of errors.

By default, the data monitors only show the most recent athmospheric data. The monitors shift to the left over the course of time. Using mouse interaction, the user may scroll to previous data points. The resolution of the display is adapted to the range that applies at the very moment. The pressure will be displayed in kPa, whereas the temperature values are rated in Celsius, alternatively Fahrenheit, as chosen by user.

We have acquired a satellite image of the campus of Julius Maximilian University of Würzburg, including the very precise longitudinal and latitudinal coordinates. Using linear interpolation, we can locate the Cansat very accurately on the image, to give the user a good intuition on where the device is currently situated.

6 Conclusions and future work

In the first phase, we have discussed analysed objectives, requirements and tasks of the CanSat project. A CanSat is a device that can be launched to a fairly unknown environment to monitor the specific environment properties and acquire the information of certain sensors, such as GPS, temperature and pressure data. Then the CanSat attempts to send the information to groundstation through radio link. When groundstation receives the information from CanSat, it processes and analyses data stream in order to get useful data to display modules. Then the groundstation system displays the information from CanSat in appropriate graphical format, as well as stores the stream on the harddrive.

In this report, we have designed the system structure in graphic and have defined the interface and mechanic of hardware board and located the modules on the board of CanSat. We divided the groundstation software into two parts. One is to process and store data stream, including package command, encode the acquired data and transmit the data. Another is to display the acquired data from CanSat. There are two parts of hardware, groundstation tranceiver and CanSat hardware. The control algorithm should be implemented by hardware programming for a programmed macrocontroller. Basically, to run the software of CanSat need the basic software modules including bootloader, timer, watchdog. To acquire the sensors data, the software of CanSat must interact with the hardware of sensors. There must need macro layer drivers to acquire sensor data. To run the whole system reliably and in real time, a main and robust programme module is very important for running and communication of the CanSat. In the second phase, we want to implement these parts of the whole system. We will complete the circuits of groundstation and CanSat, and then solder the components and the board. Then we need to test the hardware to insure the accuracy. After that, we will test the hardware. Meanwhile, we will develop the software of groundstation and CanSat and debug and test the software separately. Next to these activities, we will finalize the communication protocol, make modifications to the circuit, and in particular adjust the port distribution of macro-controller according to the requirement of software. To consider the security of the system data transmission, we consider to use simple encrpytion techniques in communication process. Furthermore, we will enable the groundstation software to monitor and interact with multiple CanSats distinguished by their ID address.

7 Appendix

We prefer to give the circuit layout on separate pages. The following pages show the circuit of the groundstation, the circuit of the cansat device without the microcontroller referencing the pins of the ATmel128 crumb. The last page shows the microcontroller.

References

[ATmega128] *8-bit AVR Microcontroller with 128K Bytes In-System Programmable Flash*, Atmel Corporation, 2004

[CP2102] *Single-Chip USB to UART Bridge*, Silicon Laboratories, 2004

[FM75] *Low Voltage 2-Wire Digital Temperature Sensor with Thermal Alarm*, `www.fairchildsemi.com`

[GR213] *HOLUX GR-213, GPS Receiver, User's guide*, HOLUX Technology Inc., 2005

[Kr00] Guido Krüger: *Go To Java 2*, Handbuch der Java-Programmierung, 2. Auflage, Addison-Wesley, 2000

[JP7T] Fadil Beqiri: *JP7-T Family GPS-Receiver Hardware description*, FALCOM GmbH, `www.falcom.de`

[MAX3232] $3.0V$ *to* $5.5V$*, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four* $0.1\mu F$ *External Capacitors*, Maxim Integrated Products, 1999

[MMA7260Q] $\pm 1.5g - 6g$ *Three Axis Low-g Micromachined Accelerometer*, Freescale Semiconductor, Inc., 2005, `www.freescale.com`

[MPX4115A] *Integrated Silicon Pressure Sensor for Manifold Absolute Pressure, Altimeter or Barometer Applications*, Motorola, Inc., 2001, `www.freescale.com`

[RT433F4] *Infoblatt 433MHz-FM-Mehrkanal-Transceiver*, Ingenieurbüro für Elektronik, Wildpoldsried, 2004, `www.funkmodul.com`

[TS2940] *1A Ultra Low Dropout Fixed Positive Voltage Regulator*, TSC, 2003

[WQ05] Wang Hankang, Li Quancheng, Jiang Hai: *CDMA based distribution power net automatization and data transmission system*, Power Engineering Conference, 2005. IPEC 2005. The 7th International