Future Institute of Engineering

Communication and Electronics Department

“Design and implementation of CubeSat. Prototype”

Submitted in Partial Fulfillment of the Requirements B.Sc. Degree in Electrical Engineering (Communication & Electronics)

Prepared by:

1. Ahmed Ehab Ebrahim

2. Eslam Mohamed Ali

3. Mohamed Ali Farouk

4. Mohamed Magdy Hosny

Under supervision of:

Dr.. Mohamed Mourad Mabrook

ACKNOWLEDGMENT

We would like to express our special thanks of gratitude to our supervisor Dr. Mohamed Mourad for his valuable advices, continuous support and his kind supervision.

We should transfer our special thanks to “Egyptian Space Agency” EGSA, for their summer training , selecting and supporting our project to be one of EGSA university satellite training program 2019/2020.

Many Thanks to supervisor team from EGSA, Eng. Ahmed Farag and Eng. Hader Tantawy who gave us the golden opportunity to do this challengeable project on the topic “CubeSat” which also helped us in doing a lot of researches and are came to know about so many new things we are really thankful to them.

Finally, we need to give our thanks to great staff of Future Institute of Engineering, specially Dr. Rania Foad head of communication department for her actual and continuous support.

ABSTRACT

This project is a new tool for education and demonstrations. It can be used in a classroom or training setting to introduce the basics of cube satellites, or it can be a stepping stone in a project to build and launch an actual CubeSat into space. Which demonstrates all the major subsystems of a satellite.

The project includes a definition, declaration and design of CubeSat prototype including on board computer subsystem, ADCS subsystem, Payloads in addition to ground station simulator. The project guaranteed sending and receiving data between ground station and satellite with high accuracy due to the used error checking mechanism.

This document describes a prototype of the CubeSat which provides similar functionality to the original CubeSat.

TABLE OF CONTENTS

List of figures

LIST OF TABLES

LIST OF ACRONYMS/ABBREVIATION

ADCS	Attitude and Determination Control System
COTS	Commercial-off-the-shelf
CRC	Cyclic Redundancy Check
C&DH	Command and Data Handling
EPS	Electrical Power System
GPIO	General Purpose Input/output
GEO	Geostationary
LEO	Low Earth Orbit
NASA	National Aeronautics and Space Administration
OBC	Onboard Computer
PC	Personal Computer
SCL	Serial Clock Line
SDA	Serial Data
SPI	Serial Peripheral Interface
TC	Timer Counter
TCCLKS	Timer Counter Clock Select
TT&C	Telemetry Tracking and Command
TWI	Two Wire Interface
UART	Universal Asynchronous Receiver Transmitter
USART	Universal Synchronous Asynchronous Receiver Transmitter
WDT	Watch Dog Timer
DVB-RCS	Digital Video Broadcasting-Return Channel over Satellite
GES	Ground Earth Stations
MES	Mobile Earth Stations
GADSS	Global Aeronautical Distress & Safety System
MSC	Mobile Satellite Communication
NCS	Network Coordination Stations
NCC	Network Control Center

SCC	Satellite Control Center
SCS	System Control Segment
OSN	Operational Support Network
NCS	Network Control Station
MCC	Mission Control Centers
FOSAT	Fiber Optic Satellite
TDRSS	Tracking and Data Relay Satellite System
GSTDN	Ground Spaceflight Tracking and Data Network

Chapter One

1 INTRODUCTION

A satellite is an object in space that orbits or circles around a bigger object[1] There are two kinds of satellites: natural (such as the moon orbiting the Earth) or artificial (such as the International Space Station orbiting the Earth). But usually when someone says "satellite" they are talking about a "man-made" satellite. Man-made satellites are machines made by people. These machines are launched into space and orbit Earth or another body in space. Artificial satellites, however, did not become a reality until the mid-20th century. The first artificial satellite was Sputnik, a Russian beach-ball-size space probe that lifted off on Oct. 4, 1957. Thousands of artificial, or man-made, satellites orbit Earth. Some take pictures of the planet that help meteorologists predict weather and track hurricanes. Some take pictures of other planets, the sun, black holes, dark matter or faraway galaxies. These pictures help scientists better understand the solar system and universe[1].

Figure 1.1: The Jason-2 satellite orbits Earth. It carries tools and sensors to help scientists study the oceans. [1]

1.1 SATELLITE TECHNOLOGY AND DEVELOPING COUNTRIES

Technology that is enabled by Satellites and Satellite applications have the potential to meet **significant** needs in developing countries "including food production". In fact, there is a strong relationship between development, well-being of the citizens and space activities.

Satellite based technology that can be of use in developing countries:

1 **Remote sensing**:
   1- Urban Planning: High resolution satellite imagery can provide the planners with needed information about the growth of the city.
   2- Food Security: Satellites can improve food production for starving people by improving information available to the agricultural sector.

2 Communication:
   1- Satellites can improve access to medical care (by enabling telemedicine or transmitting valuable health-related data), government services and economic efficiency.

   2- Satellite based communications can provide improvements to formal and informal opportunities by distance education for example. Also, television, radio or internet for informal education.

3 Navigation:
   1- **Aviation**: civil aviation can be a valuable source of economic growth.

   2- **Wild life tracking**: wildlife in some developing countries it is a valuable natural resource and key challenge in many countries to manage it since many species are endanger due to human activities. Management can be facilitated using satellite based wildlife tracking. (special designed GPS receivers attached to animals without harm or impeding).[2]

1.2 SATELLITE OVERVIEW

Satellites transmit and receive signals. The signal is sent from a station on the Earth's surface, the satellite receives the signal and rebroadcasts it to other places on the earth (television, telephone, fax, Internet).

1.2.1 Satellite system components

A satellite communication system consists of mainly two segments; space segment and ground segment. Therefore, accordingly there will be two types of subsystems namely, space segment subsystems and ground segment subsystems.

- **Space segment:** includes the satellite and ground facilities (the tracking, telemetry and command (TT&C)).

- **Ground segment:** ground segment performs mainly two functions. Those are transmission of a signal to the satellite and reception of signal from the satellite. Ground control station, Data reception station, and/or user terminal are the major subsystems that are present in earth segment.

1.2.2 Types of Satellites and Applications

Satellite plays a vital role in our daily life. Followings are the most common applications of satellite communication; Radio broadcasting and voice communications, TV broadcasting such as Direct to Home (DTH), Military applications and navigations, Remote sensing applications, Weather condition monitoring & Forecasting.

Artificial satellites vary in size and cost depending on the goal it was created for. They can be small enough to fit in the palm of your hand or as huge as the International Space Station (ISS).

Satellite types according to mass:

- Large satellites: More than 1,000 kg
- Medium-sized satellites: 500-1,000 kg
- Small satellites:
    - Minisatellite: 100-500 kg
    - Microsatellite: 10-100 kg
    - Nanosatellite: 1-10 kg
    - Picosatellite: Less than 1 kg .[3]

1.3 CUBESAT OVERVIEW

Nanosatellites are loosely defined as any satellite weighing less than 10 kilograms. CubeSats must also comply with a series of specific criteria that control factors such as their shape, size and weight.

CubeSats can come in various sizes, but they are all based on the standard CubeSat unit, namely a cube-shaped structure measuring 10x10x10 centimeter with a mass of somewhere between 1 and 1.33 kg. This unit is known as 1U. After the first few years, this modular unit was multiplied and larger nanosatellites are now common (1.5U, 2U, 3U or 6U). The very specific standards for CubeSats help reducing costs. The standardized aspects of CubeSats make it possible for companies to mass-produce components and offer off-the-shelf parts.

As a result, the engineering and development of CubeSats becomes less costly than highly customized small satellites. The standardized shape and size also reduces costs associated with transporting them to, and deploying them into space.

CubeSats began as a collaborative effort in 1999 between Jordi Puig-Suari, a professor at California Polytechnic State University (Cal Poly), and Bob Twiggs, a professor at Stanford University's Space Systems Development Laboratory (SSDL). The original intent of the project was to provide affordable access to space for the university science community, and it has successfully done so [3].

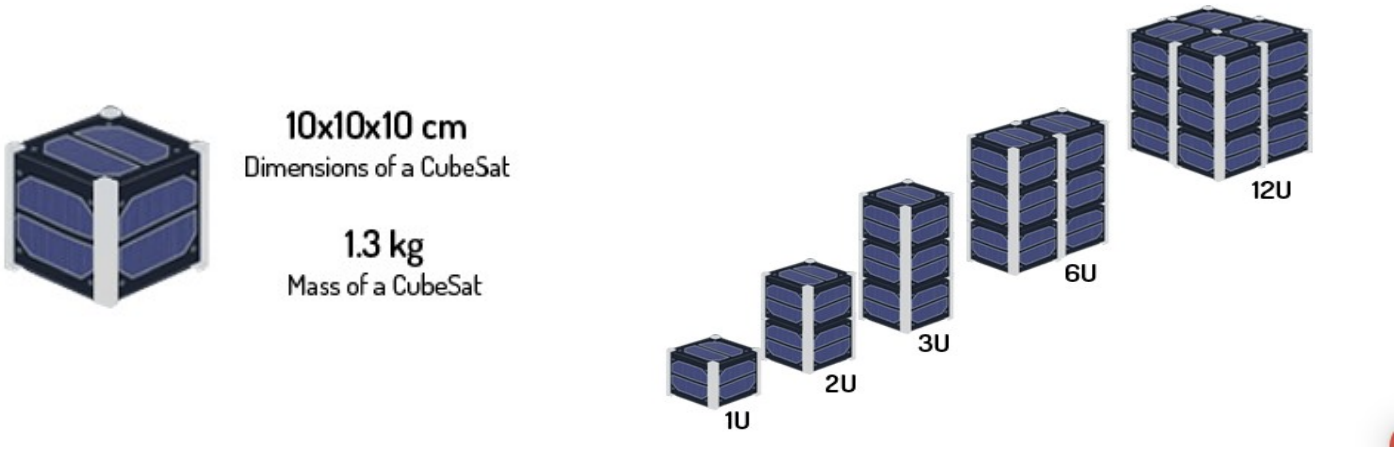


Figure 1.2: The CubeSat standard

1.3.1 What are the main advantages of CubeSats?

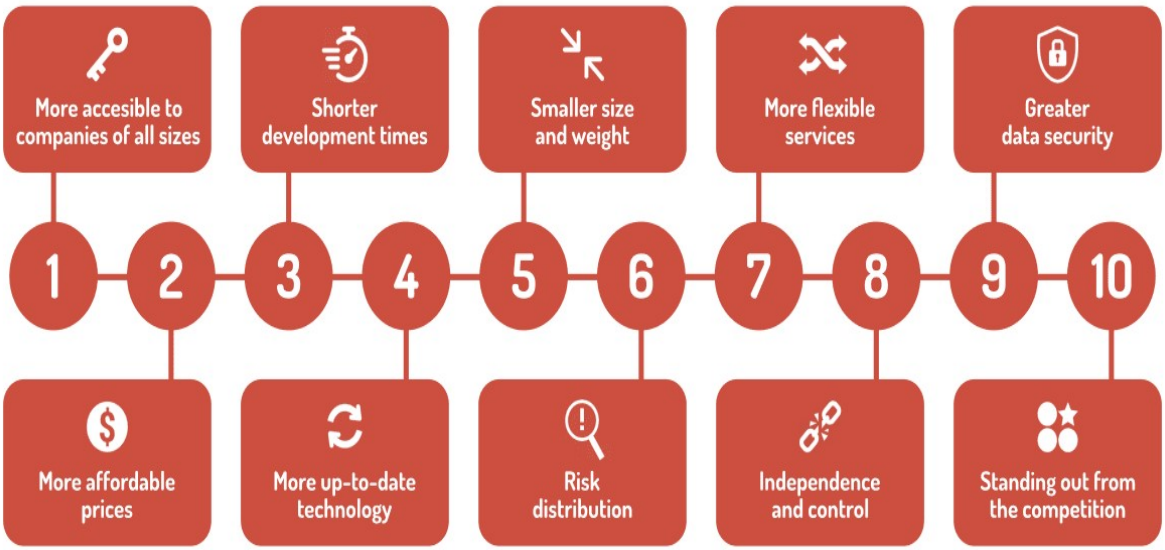


Figure 1.3: advantages of CubeSat.[3]

1.3.2 CubeSat architecture and subsystems

In its very basic form, a CubeSat's architecture is similar to what is presented in Figure 1.4. The OBC is at the center of all communications between other subsystems via a serial bus interface. it has the additional function to record housekeeping parameters and telemetry payload data collected at given timestamps or coordinates, before initiating the transmission to the ground station during an overpass. Each subsystem is assigned a dedicated task.

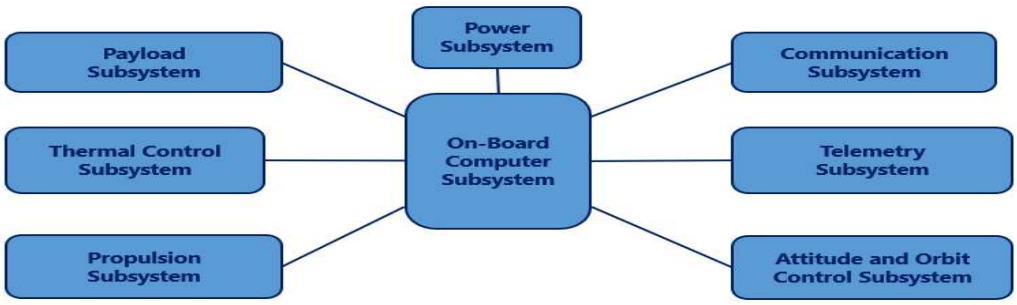


Figure 1.4: CubeSat architecture and subsystems

There is a module that is responsible for feeding the entire satellite peak, called **power**, another is responsible for determining and controlling the attitude of the satellite (**ADCS**), the subsystem whose main mission is to establish the communication link with the segment of Earth is the call of **communications**, there is usually an **OBC** system, which is responsible for managing information between systems and in many cases also handles the payload of the system (**Payload**).

1.4 GOALS OF THE CUBESAT

The main goal of this project is to replicate as much as possible the functionality of the original CubeSat. We have tried to use materials and utilize software and general-purpose hardware as much as possible.

The other goals are listed below:

• Make all hardware and software freely available through open source.

• Have a robust design that makes maintenance simple.

• Have functionality that emulates a real CubeSat.

• Have a modular design that allows different subsections to be swapped out for alternatives.

We cooperated with the **Egyptian Space Agency** engineers, in a collaborative effort with the industry, to design and build the CubeSat "educational satellite system".

CubeSat project enables us to teach space systems engineering by giving students the opportunity to:

1. Verify each subsystem against a set of design requirements.
2. Integrate the entire system.
3. Perform system-level verification and validation procedures.

Chapter Two

2 CUBESAT DESIGN AND IMPLEMENTATION

2.1 INTRODUCTION

This chapter covers the detailed design and implementation of the CubeSat. Each component of the CubeSat subsystems is analyzed individually and schematic diagrams of each subsystem are also included.

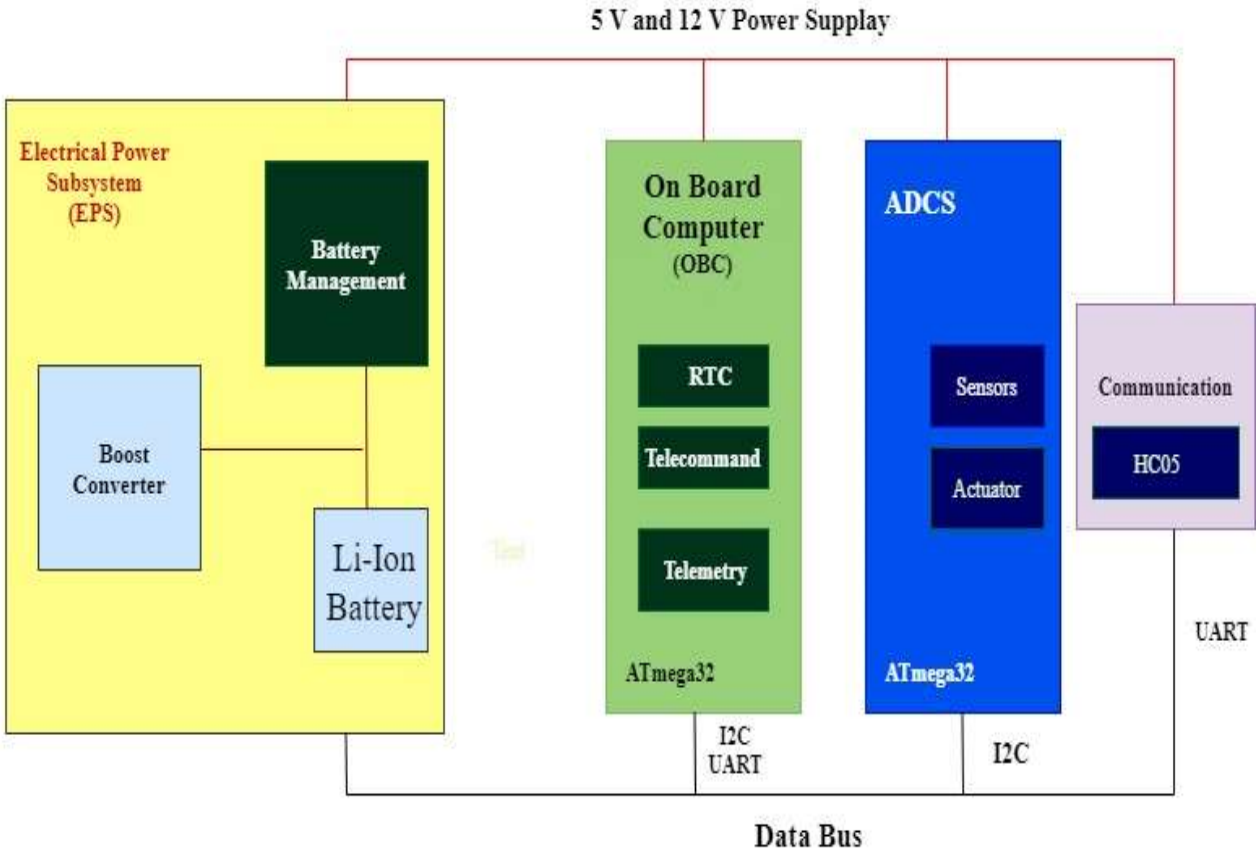


Figure 2.1: CubeSat Subsystem Block Diagram

2.2 ON BOARD COMPUTER (OBC)

For any satellite, the OBC module is considered the brain of the satellite, is responsible for managing and directing the various resources of the system. It also has the task of intercommunicating the different modules. Although each module has some independence, it is also obliged to report certain variables to the central OBC module, as well as having the capacity to make requests either from resources or from information from other modules [4].

In addition to performing inter-modular tasks, in some cases it is required that the OBC module perform other tasks. They can be arithmetic calculations to free up load on other modules, as well as perform storage tasks.

OBC controls all aspects of the CubeSat. It collects data on the operation and status of the CubeSat, manage the information requested and deliver all subsystems within the CubeSat, formats the telemetry and transmits it to the Ground Station for display and analysis. The protocol that is used inside the satellite, is I2C.

A block diagram showing all different components of the OBC is represented in Figure2.2.

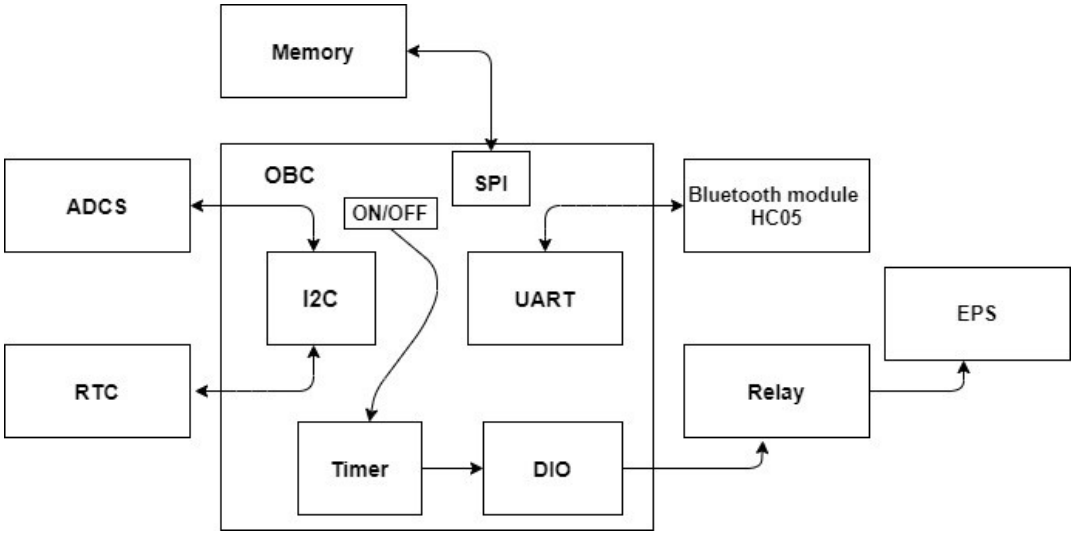


Figure 2.2: Detailed view of bus connections and OBC main component

2.2.1 The different specific roles assigned to the OBC

- Collect housekeeping and report telemetry
- Telemetry storage in memory
- Encoding of telemetry
- Forward telemetry to ground control station
- Satellite autonomous control and monitoring (time tagged commands).
- activation / deactivation of other subsystems depending on certain conditions
- decoding telecommands and executes them by in turn commanding the other subsystems to achieve the desired action.
- Timing reference.
- Flashing over the air.

The following sections will cover each block in detail and discuss the hardware/Software implementation associated with each section of the OBC.

2.2.2 Communication protocols

❖ **Contrast and compare Serial versus parallel data transfer**

Computers transfer data in two ways: parallel and serial. In parallel data transfers, often eight or more lines (wire conductors) are used to transfer data to a device that is only a few feet away. Devices that use parallel transfers include printers and IDE hard disks; each uses cables with many wires. Although a Lot of data can be transferred in a short amount of time by using many wires in parallel, the distance cannot be great. To transfer to a device located many meters away, the serial method is used. In serial communication, the data is sent one bit at a time, in contrast to parallel communication, in which the data is sent a byte or more at a time .[5]

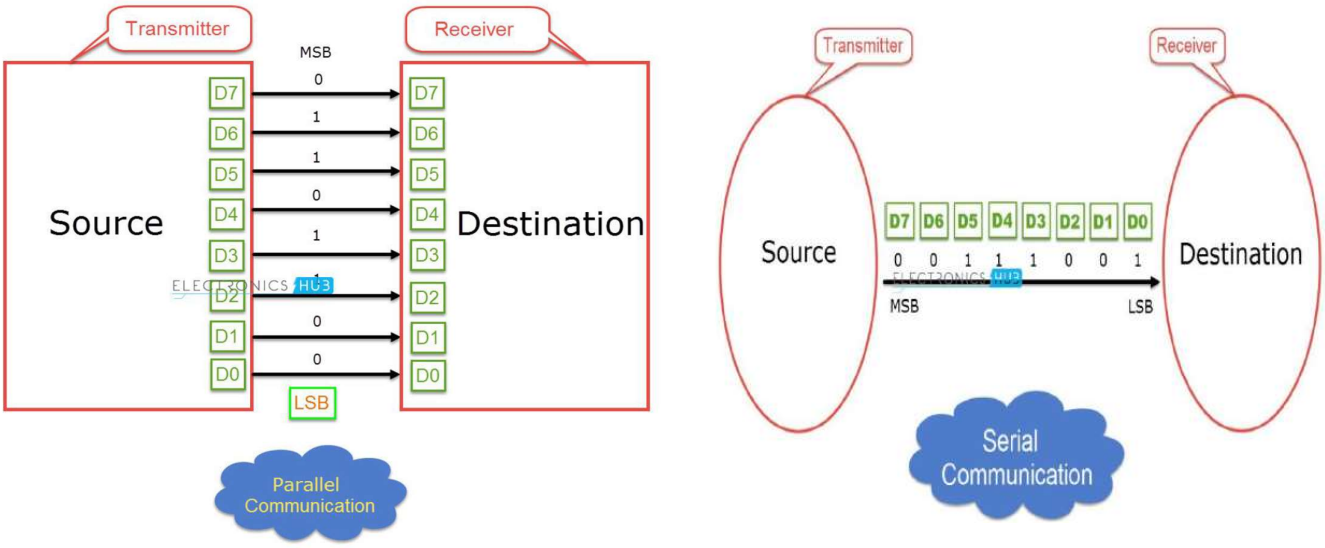


Figure 2.3: Serial versus parallel data transfer

❖ **Synchronous Vs Asynchronous**

Serial data communication uses two methods, asynchronous and synchronous. The synchronous method transfers a block of data (characters) at a time, whereas the asynchronous method transfers a single byte at a time.

Synchronous communication:

• Transmitter & receiver use common clock

• Used for short distance, high volume, in blocks (instead of individual characters) and fast transfer.

- Synchronous serial communication interfaces: I2C, SPI, Microwire, USB, IEEE1394 (FireWire).[5]

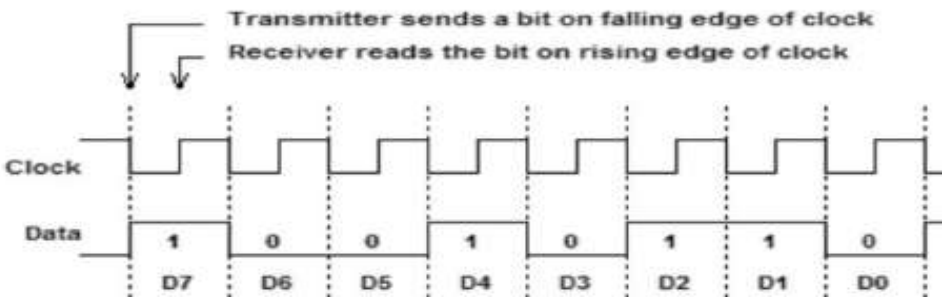


Figure 2.4: Synchronous communication [6]

Asynchronous Communication:

- Framing: Each data frame consists of a start bit, a variable number of data bits, an optional parity bit, and 1 or 2 stop bits. The most common configuration is 1 start bit, 8 data bits, no parity bit, and 1stop bit ("8N1").
- Character oriented
- Standard communication speeds: • 110, 150, 300, 600, 1200, 2400, 4800, 9600 ,……..., 115.2 kbaud [6]

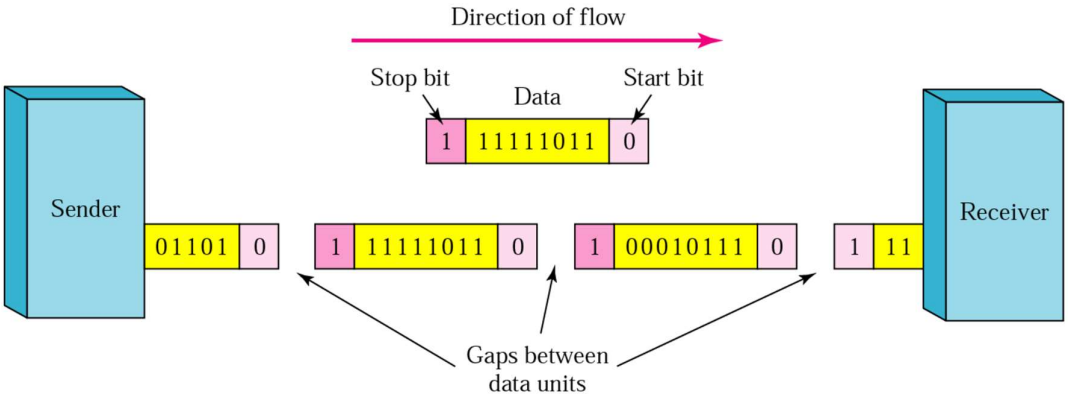


Figure 2.5: Synchronous communication .[6]

It is possible to write software to use either of these methods, but the programs can be tedious and long. For this reason, special IC chips are made by many manufacturers for serial data communications. These chips are commonly referred to as UART (uni-versal asynchronous receiver-transmitter) and USART (universal synchronous-asynchronous receiver-transmitter). The AVR chip has a built-in USART

2.2.2.1 Universal Asynchronous serial Receiver and Transmitter (UART).

The need for serial exchange of data in a CubeSat represents a major requirement. This type of transmission allows for one bit to be transmitted or received at a time between two devices and involves the transfer of data over a single wire for each direction. The general operation of serial

interfaces consists of converting parallel data from a microcontroller to a serial bit stream and vice versa .

UART Features :

- Full Duplex Operation (Independent Serial Receive and Transmit Registers).
- High Resolution Baud Rate Generator
- Supports Serial Frames with 5, 6, 7, 8, or 9 Data Bits and 1 or 2 Stop Bits
- Odd or Even Parity Generation and Parity Check Supported by Hardware
- Framing Error Detection
- Three Separate Interrupts on TX Complete, TX Data Register Empty, and RX Complete. [7]

We use UART to Make a communication channel between OBC (ATmega32) and PC (GCS) via Bluetooth. The ATmega32 has one UART/USART which is incorporated on-chip.

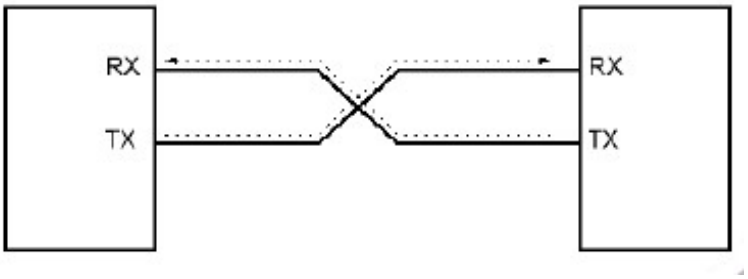


Figure 2.6: UART Receiver and Transmitter

The UART is initialized by configuring control registers that determine the baud rate, parity, number of stop bits.

```
void USART_init(void)
{

    /*           Set Baud Rate                    */
    UBRRH = (uint8_t)(BAUD_PRESCALLER>>8);
    UBRRL = (uint8_t)(BAUD_PRESCALLER);

    /*    Enable Receiver and Transmitter         */
    UCSRB = (1<<RXEN)|(1<<TXEN)|(1<<RXCIE);

    /* Set Frame Format  8-bit data 1 Stop bit */
    UCSRC = (1<<UCSZ0)|(1<<UCSZ1)|(1<<URSEL);
}
```

Figure 2.7: UART initialization

- **Baud Rate:** is the "symbol rate" of the transmission system ,For a UART, baud rate is same as the number of bits per second (bps)

- **Bit Duration**
  - Each bit is transmitted for a fixed duration.
  - The duration must be known to Tx and Rx
  - Baud rate (f) determines the duration (T)
  - Baud rate is the number of transitions per second a Typically measured in "bits per second (bps)"
  - T = 1/f . F = 9600 baud. T = ~104.17 μs.

❖ Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$\text{BAUD} = \frac{f_{\text{OSC}}}{16(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{\text{OSC}}}{16\text{BAUD}} - 1$
Asynchronous Double Speed mode (U2X = 1)	$\text{BAUD} = \frac{f_{\text{OSC}}}{8(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{\text{OSC}}}{8\text{BAUD}} - 1$
Synchronous Master mode	$\text{BAUD} = \frac{f_{\text{OSC}}}{2(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{\text{OSC}}}{2\text{BAUD}} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD	Baud rate (in bits per second, bps).
f_{osc}	System oscillator clock frequency.
UBRR	Contents of the UBRRH and UBRRL Registers, (0-4095).

Table 1: Baud Rate Register Setting [7]

$$Error\left[\%\right] = \left(\frac{\text{BaudRate}_{\text{Closest Match}}}{\text{BaudRate}} - 1\right) \times 100\,\%$$

Baud Rate [bps]	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X= 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
Baud Rate [bps]	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X= 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%

Table 2: Examples of UBRR Settings for Commonly Used Oscillator Frequencies.[7]

Function Name	**Description**
USART_init	To initiate the UART by configuring control registers that determine the baud rate, parity, number of stop bits.
USART_receive	To receive byte/char
USART_send	To Transmit byte/char
USART_putstring	To Send string
UART_RxString	To receive string

Table 3: UART Driver functions

2.2.2.2 HC05 Bluetooth module

HC-05 is a Bluetooth device used for wireless communication. It works on serial communication (USART). It is a 6-pin module.
The device can be used in 2 modes; data mode and command mode.

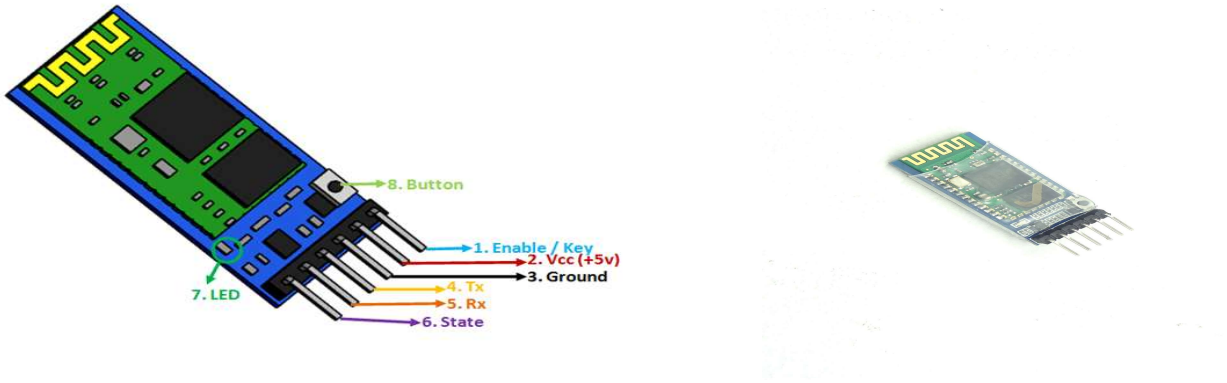


Figure 2.8: HC05 Bluetooth Module [8]

HC05 is the **communication subsystem** of our CubeSat has two main purposes:

- Transmit telemetry data.
- Provide a mean for the satellite to communicate with ground station and vice versa.

The data/command that is transmitted or received from OBC/PC to OBC/PC via Bluetooth takes special format. The transmitter encodes the data and receiver decode it. The data frame that used in this project is shown below, this format enables us to encrypt the data that is transmitted / received.
Each block of data is always appended with cyclic redundancy check (CRC) bits which helps in detecting errors by means of sending a positive or negative acknowledgement.

ID	Command	Subcommand	NMI	NMO	Data	CRC
1byte	1byte	1byte	1byte	1byte	variable	1byte

Table 4: Data / Command format. (data encoding)

2.2.3 Inter-Integrated Circuit (I2C)

The protocol that is used inside the CubeSat, is I2C. The data transfer between subsystems is made possible by the OBC's microcontroller's integrated serial interfaces (I2C).

❖ **What is I2C?**

- The name stands for "Inter -Integrated Circuit"
- A Small Area Network connecting ICs and other electronic systems
- The IIC is also referred to as I2C (I2C) or I square in many technical literatures.
- The I2C bus was originally started by Philips, Now It is a widely used standard adapted by many chip making companies
- I2Cis ideal for communication between low-speed devices for a reliable communication over a short distance.
- Today, a variety of devices are available with I2C Interfaces
  - Microcontroller, EEPROM, Real-Timer, interface chips, LCD driver, A/D converter.[6]

The Two-wire Serial Interface (TWI) or (I2C) is ideally suited for typical microcontroller applications. The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for

data (SDA). The only external hardware needed to implement the bus is a single pull-up resistor for each of the TWI bus lines. All devices connected to the bus have individual addresses, and mechanisms for resolving bus contention are inherent in the TWI protocol. [7]

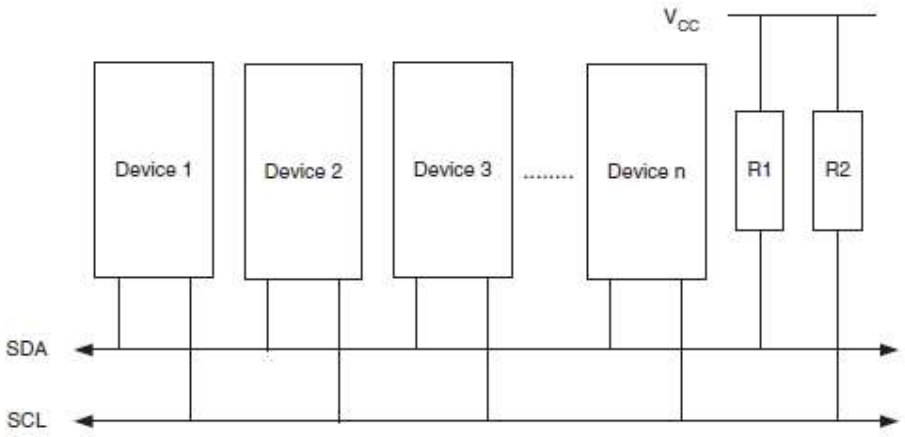


Figure 2.9 : TWI Bus Interconnection .[7]

❖ Features

- Simple Yet Powerful and Flexible Communication Interface, Only Two Bus Lines Needed.
- Both Master and Slave Operation Supported
- Device Can Operate as Transmitter or Receiver
- 7-bit Address Space allows up to 128 Different Slave Addresses
- Multi-master Arbitration Support
- Up to 400kHz Data Transfer Speed
- Noise Suppression Circuitry Rejects Spikes on Bus Lines
- Fully Programmable Slave Address with General Call Support

Term	Description
Master	The device that initiates and terminates a transmission. The master also generates the SCL clock.
Slave	The device addressed by a master.
Transmitter	The device placing data on the bus.
Receiver	The device reading data from the bus.

Table 5: TWI Terminology [7]

❖ **Data Transfer and Frame Format**

Transferring Bits : Each data bit transferred on the TWI bus is accompanied by a pulse on the clock line. The level of the data line must be stable when the clock line is high. The only exception to this rule is for generating start and stop conditions.

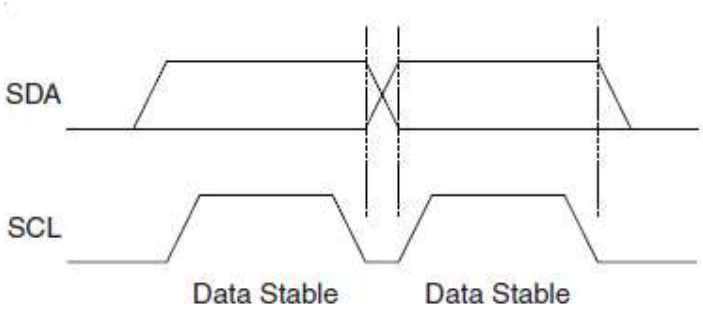


Figure 2.10: Data Change[7]

START and STOP Conditions

The master initiates and terminates a data transmission. The transmission is initiated when the master issues a START condition on the bus, and it is terminated when the master issues a STOP condition. Between a START and a STOP condition, the bus is considered busy, and no other master should try to seize control of the bus. A special case occurs when a new START condition is issued between a START and STOP condition. This is referred to as a REPEATED START condition, and is used when the master wishes to initiate a new transfer without releasing control of the bus. After a REPEATED START, the bus is considered busy until the next STOP. This is identical to the START behavior, and therefore START is used to describe both START and REPEATED START for the remainder

of this datasheet, unless otherwise noted. As depicted below, START and STOP conditions are signaled by changing the level of the SDA line when the SCL line is high.(ATmega32,data sheet)

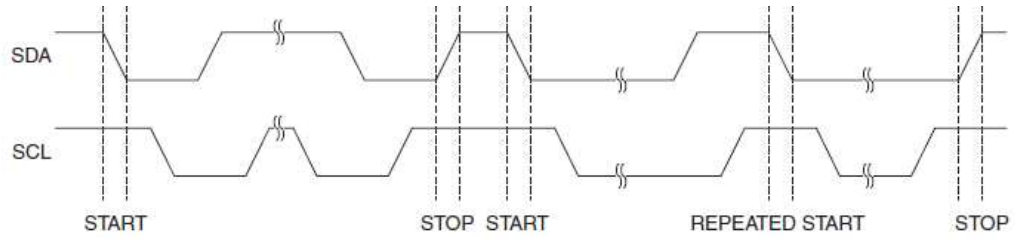


Figure 2.11: START, REPEATED START, and STOP Conditions[7]

The SCL frequency is generated according to the following equation:

$$\text{SCL frequency} = \frac{\text{CPU Clock frequency}}{16 + 2(\text{TWBR}) \cdot 4^{TWPS}}$$

- TWBR = Value of the TWI Bit Rate Register
- TWPS = Value of the prescaler bits in the TWI Status Register

❖ **Transmission Modes**

The TWI can operate in one of four major modes. These are named Master Transmitter (MT), Master Receiver (MR), Slave Transmitter (ST) and Slave Receiver (SR). Several of these modes can be used in the same application. As an example, the TWI can use MT mode to write data into a TWI EEPROM, MR mode to read the data back from the EEPROM.[7]

Function	**Description**
I2C_Master_Init	used to initialize microcontroller as a master
I2C_Slave_Init	used to initialize microcontroller as a slave
I2C_Master_T	Master transmission mode
I2C_Master_R	Master receive mode
I2C_Slave_Listen	Wait, if any master call it ,it will replay

I2C_Slave_Transmit	Slave transmission mode
I2C_Slave_Receive	Slave Receive mode
I2C_Start	Makes Start bit & make sure that the Slave address is Exist
I2C_Rep_Start	Makes Start bit(for reading) & make sure that the Slave address is Exist
I2C_Stop	used to generate I2C Stop Condition.
I2C_Write	used to send a byte on SDA line using I2C protocol
I2C_Read	used to receive a byte on SDA line using I2C protocol.

Table 6: I2C Driver functions

2.2.3.1 REAL-TIME CLOCK (RTC)

The RTC allows the OBC to have an independent time keeping system. This is important on satellites for avoiding conflicts. Implementing and synchronizing an accurate RTC onboard all the required subsystems (OBC and ground station) will result in cooperation throughout the satellite.

The DS3231 is an RTC IC developed by Maxim Integrated. It is a low cost, extremely accurate RTC IC with communication over I2C Interface.

An interesting feature of DS3231 RTC IC is that it has integrated crystal oscillator and temperature sensor and hence you don't have to connect an external crystal.

RTC module acts as slaves while a microcontroller acts as a master.

The DS3231 RTC Module used in this project is shown in the image below.

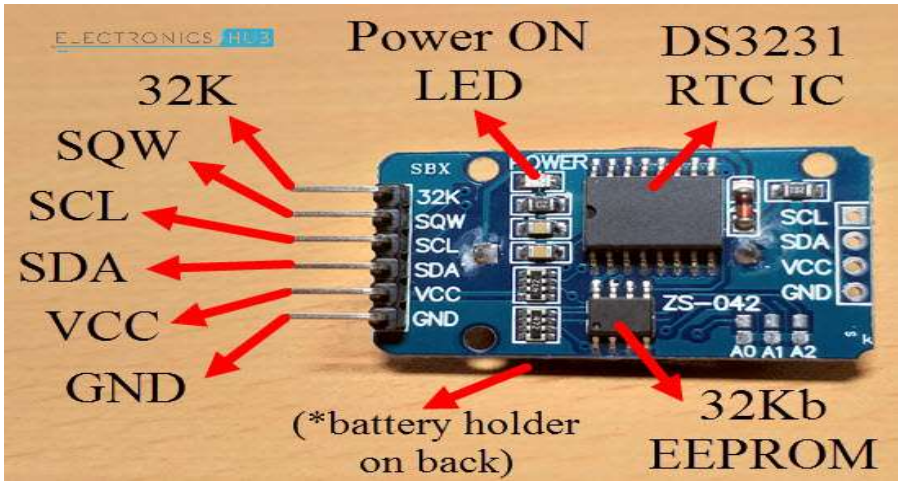


Figure 2.12: DS3231 RTC Module[9]

Function Name	Description
ds3231_init ds3231_set ds3231_get	Initializes the clock, counter and interrupt for RTC. Sets the time counter value. Get the time counter value

Table 7. RTC Driver functions

2.3 ELECTRICAL POWER SUBSYSTEM (EPS)

The EPS accomplishes the following functions:

- Supply a continuous source of electrical power to CubeSat loads during the mission life.
- Control and distribute electrical power to the CubeSat.
- Support power requirements for average and peak electrical load.

- Provide converters and regulated dc power buses.
- Provide command and telemetry capability for EPS health and status, as well as control by ground station or an autonomous system.
- Protect the CubeSat Subsystems against failures within the EPS.

2.3.1 Power Distribution :

A CubeSat 's power distribution system consists of cabling and Relays to tum power on and off to the CubeSat loads. It also includes command decoders to command specific load relays on or off.
Because the regulation requirements for these loads vary, the bus voltage may need further regulating, leveling up or down.

Figure 2.13: DC/DC Step up converter

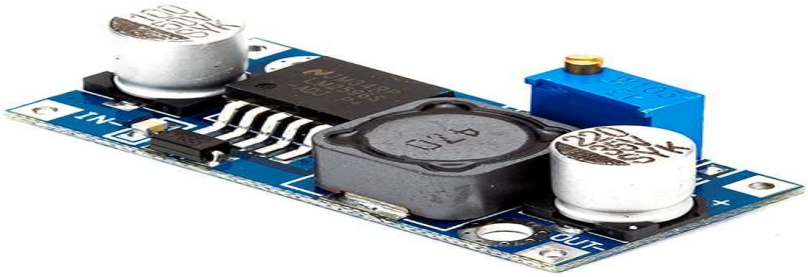


Figure 2.14: DC/DC Step down converter

We monitor the battery's voltage and regulate battery charge/discharge. Provide command and telemetry capability for EPS status, as well as control by ground station or an autonomous system.

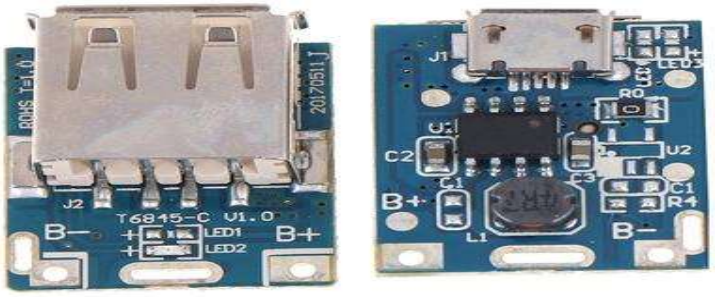


Figure 2.15: Battery charger Lithium with protection 3.7V T6845-C Module

The original CubeSat has Power **sources** are divided into primary and secondary sources.

1-**Primary Source:** The source that generates the electrical power is Solar cells which convert solar radiation into electrical energy.

Satellites orbiting Earth go through a shaded region on the other side of Earth, away from the sun, defined as eclipse. Depending on the type of orbit, eclipse lasts for a few minutes up to a few hours and during this time, the solar cells do not produce electrical power, leaving the satellite unpowered.

2-**Secondary source**: A backup source has to be available in these conditions and electrical energy needs to be stored during sun exposure in order to be used during eclipse. (banks of rechargeable- Lithium Ion -batteries).

In this project we use Lithium Ion -batteries as a power source. We will discuss LI-Ion Battery in the following section.

2.3.2 Lithium Ion -batteries

we humans have to depend on Batteries for powering our portable or remote electronic devices. The most common type of rechargeable batteries that you find in consumer electronics is Lithium ion or type.

- Li-ion Battery Chemistry and working:

As the name obviously indicates, the Lithium Ion batteries use the Lithium ions to get the job done. Lithium is a very light metal with high energy density, this property enables the battery to be light in weight and provide high current with a small form factor. Energy density is the amount of energy that can be stored in per unit volume of the battery, the higher the energy density the smaller the battery will be. Despite the overwhelming properties of lithium metal, it cannot be used as an electrode directly in the batteries since lithium is highly unstable because of its metallic nature. Hence, we use lithium-ions which more or less has the same property of a lithium metal but it is non-metallic and is comparatively safer to use.

Normally the Anode of a Lithium battery is made of Carbon and the Cathode of the battery is made using Cobalt oxide or some other metal oxide. The electrolyte used to connect these two electrodes will be a simple salt solution that contains lithium ions. When discharging the positively charged lithium ions move towards the cathode and bombard it until it becomes positively charged. Now since the cathode is positively charged it attracts negatively charged electrons towards it. These electrons are made to flow though our circuit thus powering the circuit. Similarly, while charging, the exact opposite happens. Electrons from the charges flow into the battery and hence

the lithium ions move towards the anode making the cathode to lose its positive charge.

The most commonly used Lithium Ion battery is the 18650 Cells. A typical 18650 cell is shown in the image below

Figure 2.16: 18650 Lithium Ion battery

Like all batteries the Li-ion battery also has a voltage and capacity rating. The nominal voltage rating for all lithium cells will be 3.6V, so you need higher voltage specification you have to combine two or more cells in series to attain it. By default, all the lithium ion cells will have a nominal voltage of only ~3.6V. This voltage can be allowed to go down up to 3.2V when fully discharged and go as high as 4.2V when fully charged. Always remember that discharging the battery below 3.2V or charging it above 4.2V will damage the battery permanently and might also become a recipe for fireworks.

We use Two 18650 Li-Ion batteries and monitor their status. If their voltages go down to 3.2V, they will be connected to the charger automatically and if their voltage go up to 4.2V, they will be disconnected automatically.

2.4 ADCS

2.4.1 ADCS functionality

The ADCS has a support role through the CubeSat mission. He has to maintain all other modules in an operational situation. So as an entry it takes the sensors and as an output it uses actuators.

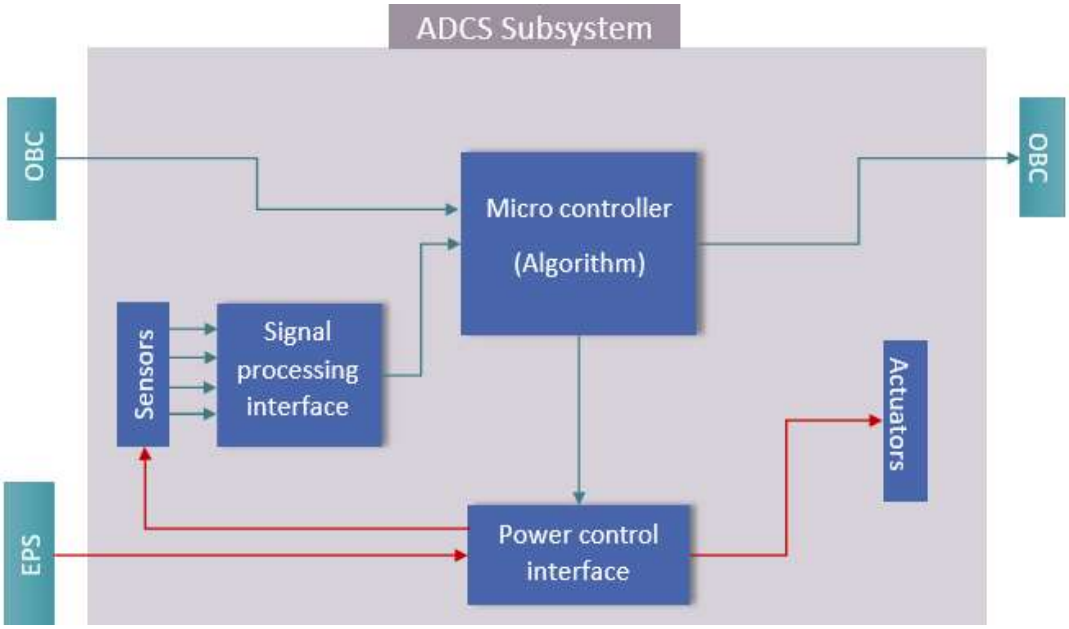


Figure 2.17 : ADCS Functioning[10]

2.4.2 ADCS Modules

The ADCS is divided into 4 modules. It is important to note that the ADCS system is currently based on a preliminary design and is subject to changes. The objectives of each module are depicted in the following list:

- The SENS is composed of a set of sensors. This set will have to harvest data in order to get information about the CubeSat position.

- The ACT are the CubeSat attitude actuators. ACT will have to adapt the CubeSat's attitude

according to the mission needs.

- The ADCS controller objectives are to collect data from sensors and to process it to get reliable positioning information. Then the ADCS will send orders to ACT in order to correct/modify the CubeSat's attitude if OBC and EPS subsystems allow it.
- The Interface module has for objective to ensure good connection with other systems of the satellite and to send data to the other systems. [10]

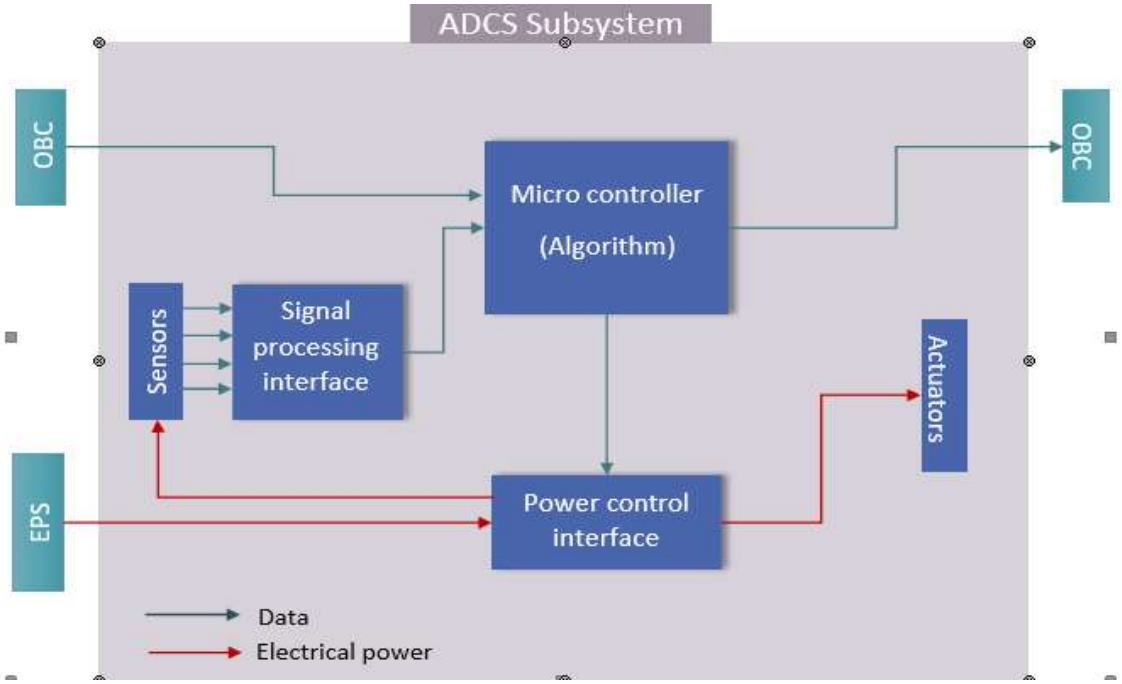


Figure 2.18: ADCS module dependence [10]

a) Sensors System (SENS)

ADCS Sensors system will be composed of absolute sensors to get constant access to the attitude relative to an external frame. And

relative sensors to get access to the current attitude relative to the previous one.

b) Actuators System (ACT)

The actuators goal is to position the CubeSat in the target attitude by rotation it around 3 axes.

- Yaw / Pitch / Roll

So the Actuators System will be placed to have control over the 3 axes (X, Y, Z).

c) Controller (CTRL)

The ADCS Controller will calculate the attitude in which the CubeSat is thanks to the data coming from Sensors. Also the Algorithm inside the controller will calculate the targeted attitude. And then will determine the rotations to accomplish for each axis.

d) Interface (INT)

The ADCS Interface is the hardware part of ADCS which transmit the signal received from Sensors to the micro-controller and it also distributes power supply coming from the EPS subsystem to the Actuators.

- Recap on ADCS

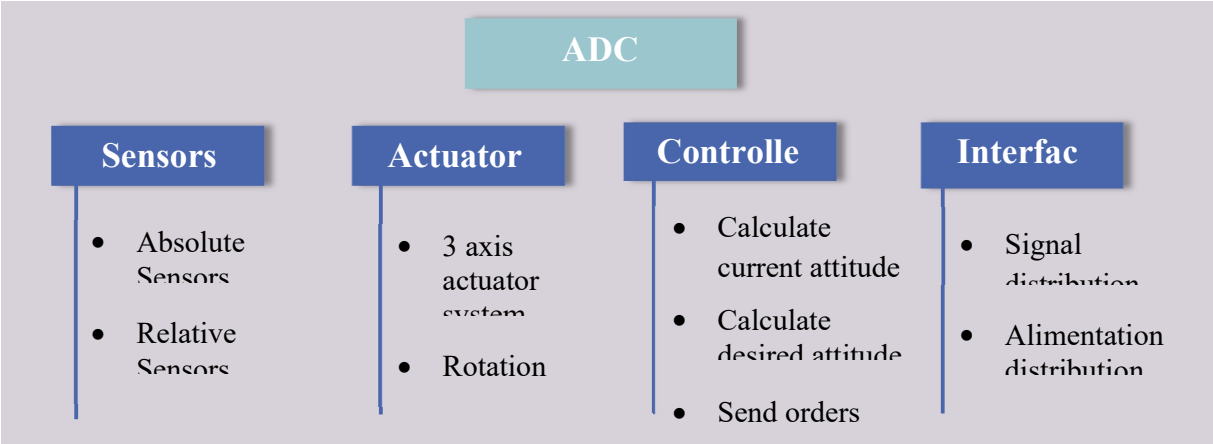


Figure 2.19: ADCS module division

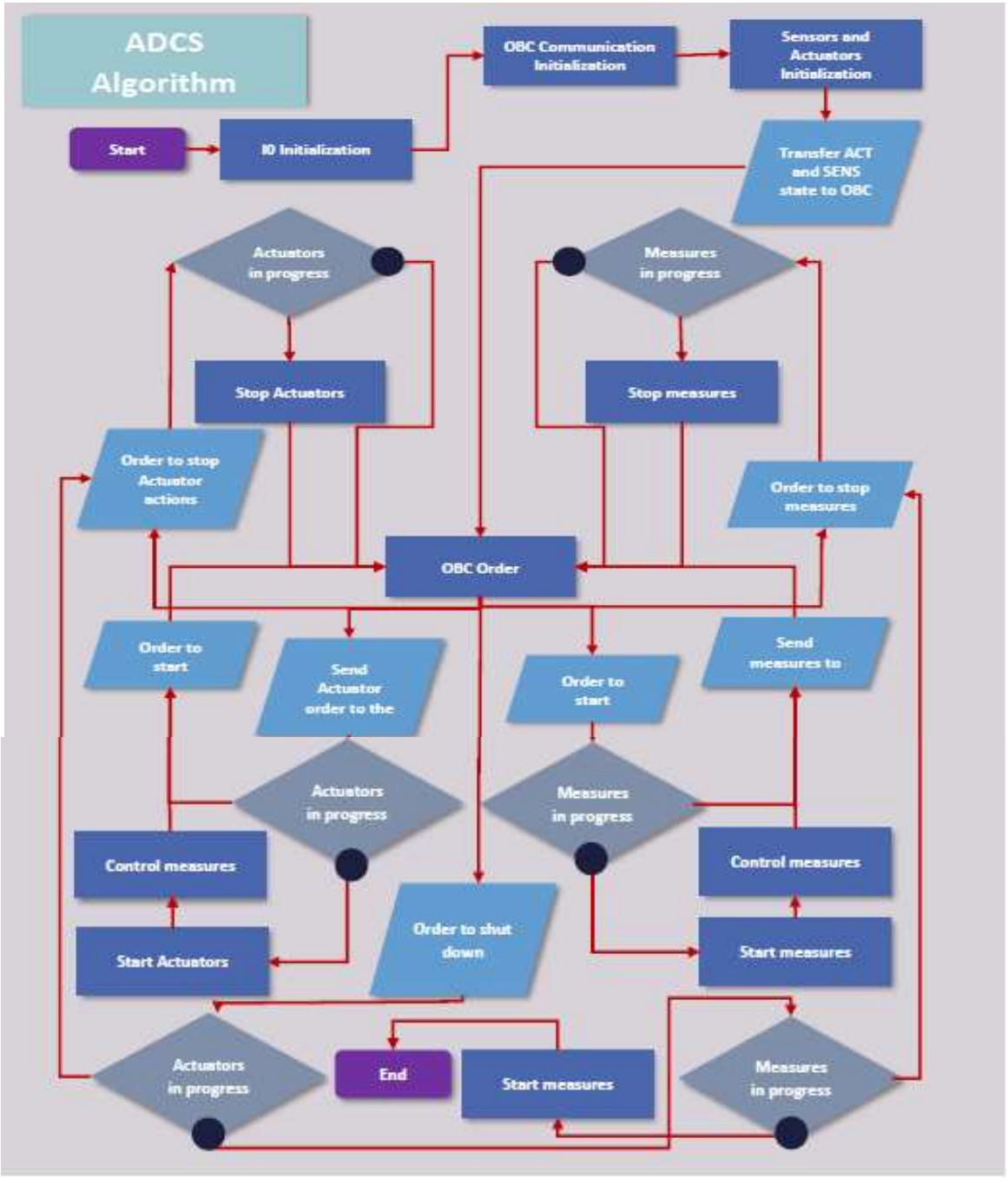


Figure 2.20:ADCS Algorithm[10]

2.4.3 ADCS Tasks

According to the previous mutual impacts of ADCS with other subsystems, ADCS has the following tasks must to be executed all over the satellite life time. That is, ADCS executing the following tasks from the moment of separation up to de-orbiting or discarding of the mission.

1. Damping the satellite angular velocity, obtained from LV after satellite
2. Attitude acquisition of the satellite where the BCS is oriented to be coincide with the assigned RCS (in Earth observation missions OCS will be this RCS). In this attitude acquisition the satellite is initially oriented towards the RCS supports the mission requirements.
3. The satellite three-axis stabilize in the RCS with the required accuracy during the imaging sessions.
4. Three-axis stabilization in nadir pointing with low accuracy during non-imaging periods
5. Attitude determination with the required accuracy during all ADCS operational modes

2.4.4 Satellite operational modes

According to the above required tasks from ADCS, the ADCS operational mode will be.

2.4.4.1 De-tumbling mode (DM)

This mode occurs after the satellite is released from the LV or after loosing of orientation due to any failure. During this mode the ADCS suppers the satellite angular velocity that received from the LV, Because of power limitation this process should be completed within specified period.

2.4.4.2 Standby Mode (SM)

the consumed power. ADCS stay in SM about 95% of the whole satellite lifetime
After DM satellite can have arbitrary attitude Automatically so after finishing DM, ADCS transfers to SM in order to make attitude acquisition of satellite (i.e. Orient the satellite BCS to be co-onside with OCS to get stabilization at nadir pointing with low accuracy) and stay in this case whenever there is no imaging tasks assigned to the satellite. In this mode the satellite attitude should be kept even with a low accuracy to avoid loosing the satellite's attitude, it is a low accuracy mode. In this mode, the most important thing is to save the system resources (i.e. lifetime of ADCS devices) and reduce

2.4.4.3 High Accuracy Mode (HAM) or Imaging Mode (IM)

In this mode, ADCS should provide the required control to achieve the pointing of the payload requirements. As an example, for imaging remote sensing satellite using magnetic actuator the satellite must be stabilized at nadir with high accuracy during imaging periods, so this mode called imaging mode (IM)..

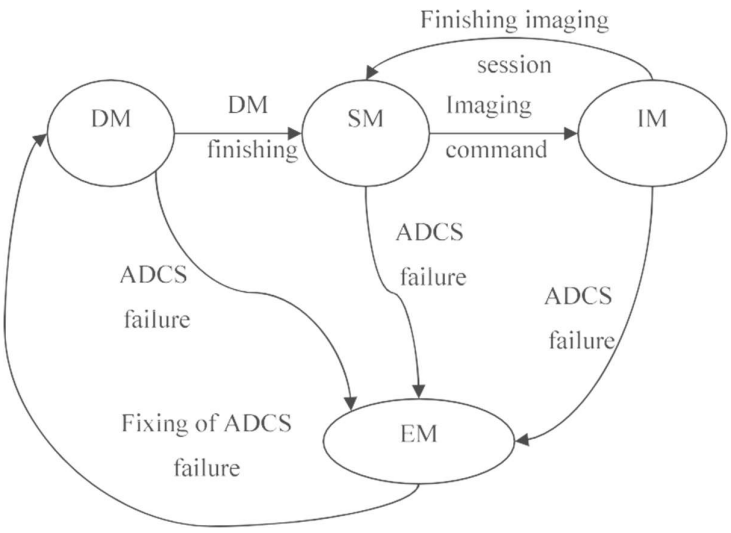
2.4.4.4 Emergency Mode

In case of any failure in ADCS (e.g. loosing satellite attitude or any failure of ADCS devices) ADCS automatically transfer to EM .In this mode ADCS switch off all ADCS devices and make diagnostic for ADCS devices according to command from ground and send TM to ground in order to take the suitable decision.

2.4.5 Transferring from one operational mode to another

The organization of transfer from one mode to another is shown in Figure (2-1).ADCS operational cyclogram and conditions for transferring between modes are as follows:

1. After separation from LV and starting of satellite operation ADCS enters DM.
2. When DM is finished, ADCS directly transfers the satellite to SM and stay in SM.
3. Before imaging time, within specified period (i.e. Period sufficient to stabilize the satellite at the required attitude with the required accuracy),ADCS transfers the satellite to IM.
4. After finishing of imaging task, ADCS transfers the satellite again to SM
5. In normal cases, the sequence of items 3-4 are repeated.
6. In case of any failure (i.e. failure in ADCS devices or attitude orientation), ADCS directly transfers the satellite to EM.



FIGUR 2.21: Organization of transferring from one operational mode to another.[11]

2.4.6 Timer in AVR ATmega32

Generally, we use timer/counter to generate time delays, waveforms or to count events. Also, the timer is used for PWM generation, capturing events etc.

- **In AVR ATmega32, there are three timers:**

- **Timer0**: 8-bit timer
- **Timer1**: 16-bit timer
- **Timer2**: 8-bit timer

➢ **Basic registers and flags of the Timers:**

- **TCNTn: Timer / Counter Register**

Every timer has timer/counter register. It is zero upon reset. We can access value or write a value to this register. It counts up with each clock pulse.

- **TOVn: Timer Overflow Flag**

Each timer has Timer Overflow flag. When timer overflows, this flag will get set.

- **TCCRn : Timer Counter Control Register**

This register is used for setting the modes of timer/counter.

- **OCRn : Output Compare Register**

The value in this register is compared with the content of the TCNTn register. When they are equal, OCFn flag will get set.[7]

- **TIMER0**

First, we need to understand the basic registers of the Timer0

1. **TCNT0: Timer / Counter Register 0**
   It is an 8-bit register. It counts up with each pulse.
2. **TCCR0: Timer / Counter Control register 0**
   This is an 8-bit register used for the operation mode and the clock source selection.

7	6	5	4	3	2	1	0
FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00

Bit 7- FOC0: Force compare match
Write only bit, which can be used while generating a wave. Writing 1 to this bit causes the wav generator to act as if a compare match has occurred.

Bit 6, 3 - WGM00, WGM01: Waveform Generation Mode

WGM00	WGM01	Timer0 mode selection bit
0	0	Normal
0	1	CTC (Clear timer on Compare Match)
1	0	PWM, Phase correct
1	1	Fast PWM

Bit 5:4 - COM01:00: Compare Output Mode

These bits control the waveform generator. We will see this in the compare mode of the timer.

Bit 2:0 - CS02:CS00: Clock Source Select

These bits are used to select a clock source. When CS02: CS00 = 000, then timer is stopped. As it [7]

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer / Counter stopped)
0	0	1	clk (no pre-scaling)
0	1	0	clk / 8
0	1	1	clk / 64
1	0	0	clk / 256
1	0	1	clk / 1024
1	1	0	External clock source on T0 pin. Clock on falling edge
1	1	1	External clock source on T0 pin. Clock on rising edge.

gets value between 001 to 101, it gets clock source and starts as the timer.

- **We use Timer 0 in ATmega32 , Using PWM mode to control The speed of The DC Motor** which is **connected** to **Reaction Wheel.**

- DC Motor's:

- Working Principle :

A simple DC motor works on the principle that when a current carrying conductor is placed in a magnetic field, it experiences a mechanical force. In a practical DC motor,the armature is the current carrying the conductor, and the field provides magnetic field.

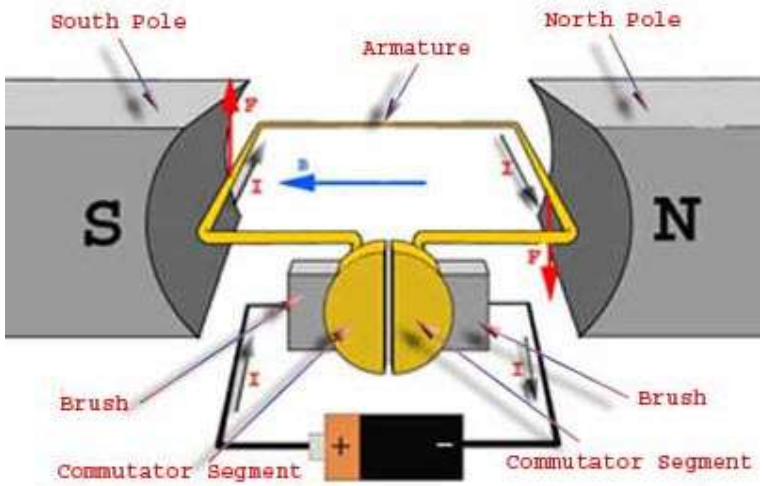


Figure 2.22 : DC Motor inner view

When the conductor (armature) is supplied with a current, it produces its own magnetic flux. The magnetic flux either adds up to the magnetic flux due to the field windings at one direction, or cancels the magnetic flux due to field windings. The accumulation of magnetic flux at one direction compared to the other exerts a force on the conductor, and therefore, it starts rotating. According to Faraday's law of electromagnetic induction, the rotating action of the conductor produces an EMF.

This EMF, according to Lenz' law, tends to oppose the cause, i.e., the supplied voltage. Thus, a DC motor has a very special characteristic of adjusting its torque in case of varying load due to the back EMF.

This implies three things:

1. Speed of the motor is directly proportional to supply voltage.
2. Speed of the motor is inversely proportional to armature voltage drop.
3. Speed of the motor is inversely proportional to the flux due to the field findings

Thus, the speed of a DC motor can be controlled in three ways:

- By varying the supply voltage
- By varying the flux, and by varying the current through field winding
- By varying the armature voltage, and by varying the armature resistance

2.4.7 Actuators

Needs:

- Physically act to modify attitude
- Compact design

a) Reaction wheel

Reaction wheels (RW) are primarily used by spacecraft for attitude control. The flywheel is attached to an electric motor, which makes it rotate when it moves. Due to the third law of newton the CubeSat will then start to counter-rotate.

Because a reaction wheel can only make the CubeSat rotate around one axis, we would need 3 of them.
Advantages:

- They are very efficient

Disadvantages:

- It has to be close to the center of mass
- Needs too much energy and space to be accurate in a CubeSat.

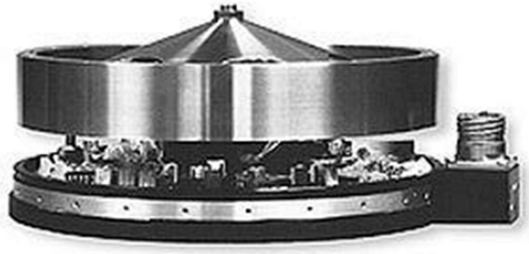


Figure 2.23: Reaction Wheel

2.4.8 Sensors:

a) Gyroscope

- Micro Electro-Mechanical (MEM) Gyroscopes:

MEMs gyroscopes have some form of oscillating component from where the acceleration and hence direction change, can be detected. This is because the conservation of motion law says that a vibrating object continues vibrating in the same plane, and any vibrational deviation can be used to derive a change in direction.

Advantage:

- Compact
- Affordable

Disadvantage:

- Noisy: drift ~0.5° per minute

Figure 2.24 : Micro Electro-Mechanical (MEM) Gyroscopes

b) Accelerometer

Accelerometers are devices that measure acceleration, which is the rate of change of the velocity of an object. They measure in meters per second squared (m/s^2) or in G-forces (g). A single G-force for us here on planet Earth is equivalent to 9.8 m/s^2, but this does vary slightly with elevation (and will be a different value on different planets due to variations in gravitational pull). Accelerometers are useful for sensing vibrations in systems or for orientation applications.

- We used in This Project MPU6050 as ACC and Gyro

- MPU-6050 IMU (3 Axis Gyro + 3 Axis Accelerometer) :

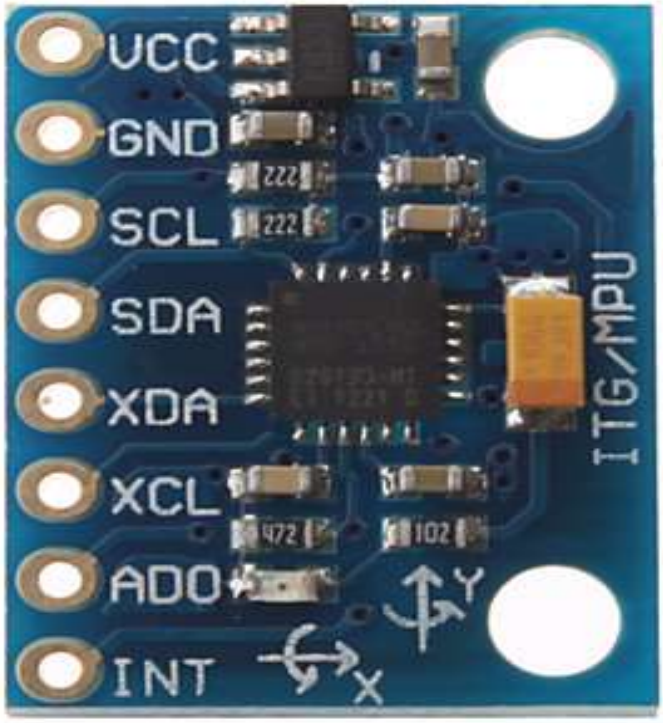


Figure 2.25: MPU6050

Features:

- I2C Digital-output of 6 or 9-axis Motion Fusion data in rotation matrix, quaternion, Euler Angle, or raw data format
- Input Voltage: 2.3 - 3.4V.
- Tri-Axis angular rate sensor (gyro) with a sensitivity up to 131 LSBs/dps and a full-scale range of ±250, ±500, ±1000, and ±2000dps.
- Tri-Axis accelerometer with a programmable full scale range of ±2g, ±4g, ±8g and ±16g

- Digital Motion Processing™ (DMP™) engine offloads complex MotionFusion, sensor timing synchronization and gesture detection.
- Digital-output temperature sensor.[12]

Hint:

We Can Add Magnetometer to achieve 9 DOF to get accurate attitude Determination.

Kalman Filter:

The Kalman filter uses mathematical method to **filter signal from noise or inaccurate measure**. It is useful to determine position or orientation even with potential measurement errors. This filter can be used to filter, smooth or predict data (past/present/future). One of its advantage is that it **provides an estimation of the error**.
In a discrete context, the Kalman filter is a recursive estimator: to estimate the current state it only needs the previous state and the current measures.
To use Kalman filter, the system **needs** to be **linearly modeled**. But if the modeling is too approximate, the filter will not be efficient enough and the estimation error will not converge fast enough.

The Kalman filter has 2 distinct states:

- Prediction (using the previous state it estimates the actual state)
- Correct (uses measurement to correct the predicted state)

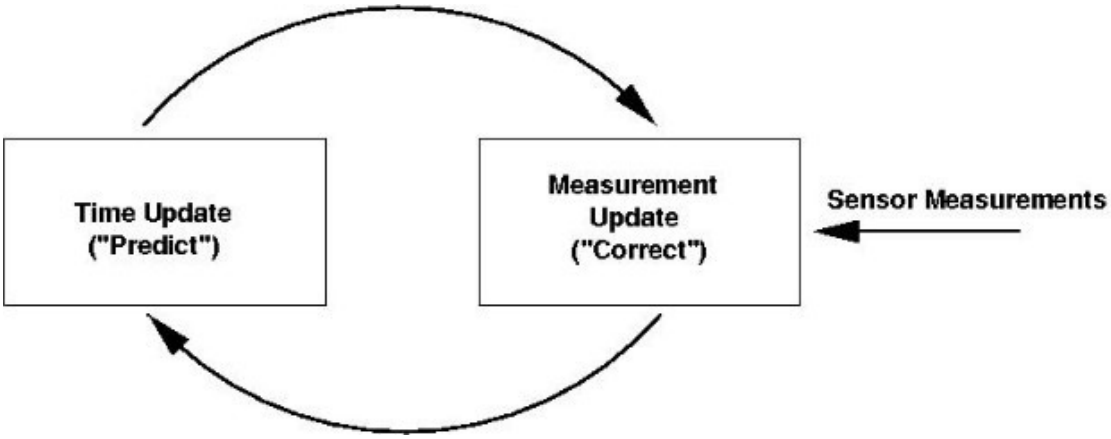


Figure 2.26: The 2 states from the Kalman Filter[10]

An extended version of the Kalman filter exist, the principal difference being the possibility to use differentiable function instead of linear (for observation and prediction).

CHAPTER 3

GROUND STATION DESIGN AND IMPLEMENTATION

3.1 WHAT ARE SATELLITE GROUND STATIONS?

Ground Station (or network of stations) is the "**Brain**" of the entire satellite network.

Ground stations are surface-based facilities which are designed to provide real-time communication with satellites. The crew at these stations send radio signals to the satellite (uplink), receive data transmissions from the satellite (downlink), and in some cases, serve as command and control centers for the satellite network.

From the ground station, data can be analyzed (or relayed to another location for analysis), the altitude and movement of the satellite and information about its critical systems can be monitored, and satellites can be controlled by decision makers.

If something goes wrong with a satellite, the crew at these stations will be the first to know. They will be charged with trying to identify the source of the problem and devise a solution.

The satellites themselves can be working fine, but it doesn't matter if no one is down here on earth to put the data they collect and transmit to use and make sure the whole system is functioning properly.

3.2 CHALLENGES AND DESIGN CONSIDERATIONS FOR SATELLITE GROUND STATIONS

The type of equipment found at a ground station will vary depending on the type of satellite and the nature of the mission, but there are similarities found in all stations.

Elements of a typical satellite ground station include:

- the system clock.
- antenna system.
- transmitting and receiving RF equipment.
- telemetry, tracking and command (TT&C) equipment.
- data-user interface.
- mission data recovery.
- station control center.

3.3 How do government-owned space agencies or private companies decide where to build their satellite ground stations?

This is determined by:

- Type of satellites being used.
- Their orbital path.
- The coverage that is required.

Oftentimes, the optimal locations for ground stations are not places where command and control or data analysis can be performed, so the mission data will have to be relayed to other ground stations or directly to end users at their location.[13]

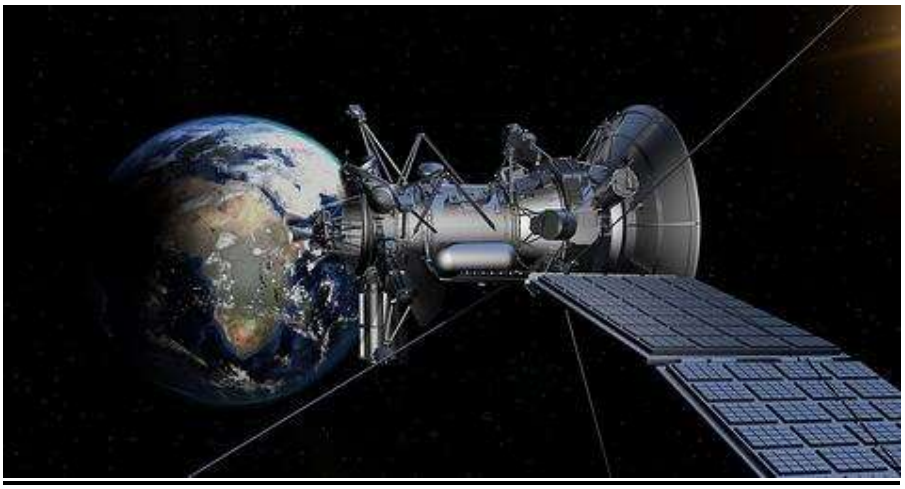


Figure 3.1: Overview of Satellite in Space[13]

- Design considerations for low earth orbit (LEO) systems are especially challenging because of their high orbital speed, typically making a full orbit every 90-110 minutes. Because of this, a ground station serving an LEO satellite can only communicate with it during the brief windows of time when the satellite is above the station's horizon plane. This means that for multiple periods throughout the day, that station has no contact with the satellite.

- Building satellites is a challenge in itself. However, ensuring that the satellite is maximizing its utility once in-orbit involves a series of complex steps and technology that needs to be accurately and efficiently deployed in order to get quality results. This requires making sure that antennae are available on the ground to constantly monitor satellite status and provide an opportunity for the operators to download the data required by their customers.[14]

Figure 3.2 :(Norcia ground station copyright ESA_events, usage licence.)[14]

3.4 Ground Communication Segment

Ground communication segment explains fixed and mobile satellite communication Ground Earth Stations with antenna systems and transmission equipment configurations for broadcasting and broadcasting via Geostationary Earth Orbit (GEO and non-GEO mobile networks). Here main components of the Ground Earth Station communication receivers and transmitters are also presented, and in particular are included Inmarsat GEO, Iridium Low Earth Orbit (OLEO), and Digital Video Broadcasting-Return Channel over Satellite (DVB-RCS) networks and Ground Earth Stations (GES). In addition, this chapter introduces ground GNSS (GPS and GLONASS), Cospas-Sarsat, and VHF and HF radio communication systems and networks. Finally, in this chapter users of radio and satellite CNS systems, equipment, and networks are described.

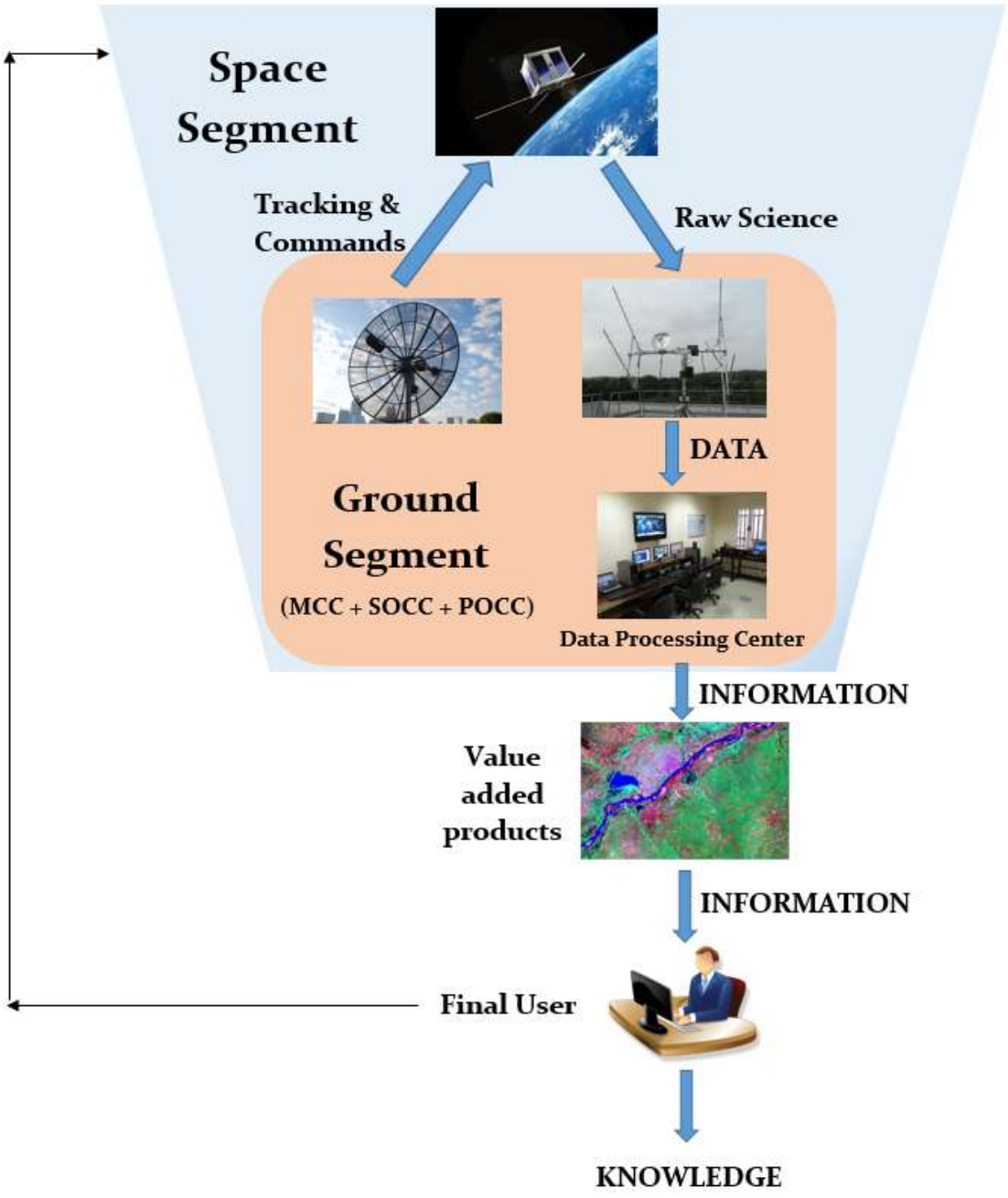


Figure 3.3 [15]
Functional Relationship Between Space Segment, Ground Segment and Final User in A Cubesat Mission

Ground communication segment is important for connecting Mobile Earth Stations (MES) via Geostationary Earth Orbit (GEO), Low Earth Orbit (LEO), or other satellite orbits and Ground Earth Stations (GES) to the terrestrial telecommunication networks (TTN). This segment consists of main integration components of the Global Aeronautical Distress & Safety System (GADSS) operational network along with space communication segment introduced in the previous chapter. During the development of the (GADSS) network, it is very important to define the ground communication segment to conduct a service together with adequate space communication segment.

The future (GADSS) satellite and radio communication, navigation, tracking, detecting, and distress alerting networks and systems can include the following integrated components:

1. Satellite communication and broadcasting subsegment.
2. Ground Earth Stations antenna subsegment.
3. Ground Earth Station Radio Frequency Equipment subsegment.
4. Ground Earth Station communication subsegment.
5. satellite communication subsegment.
6. GNSS satellite subsegment.
7. Cospas-Sarsat satellite subsegment.
8. radio communication subsegment.
9. users subsegment.

The subsegments presented in the Ground Segment have the ability to access the satellite repeater in order to provide the communication between the mobile users. Namely, Ground Segment performs mainly two functions, such as that some ground stations transmit signals to the satellite and in vice versa some of them receive those signals.

The Ground Segment containing a number of GES terminals or gateways in the Mobile Satellite Communication (MSC) is responsible for interfacing MES terminals via GEO, LEO, or other satellite Space Segments to the different type of terrestrial telecommunication systems including Internet and cellular networks. In fact, MSC networking infrastructure are arranged with the core mobile systems to which several access networks interconnect in order to allow end users to connect in the MSC network. In the Ground Segment of MSC systems the information stream that arrives through the ground infrastructure is adapted in order to be sent out on the air interface of the satellite network gateway, which in aeronautical communication systems is known as GES terminals.

The Ground Segment consists of all the ground-based elements of a spacecraft system used by operators and support personnel, as opposed to the Space Segment and user segment. It enables management of a spacecraft, and distribution of payload data and Telemetry among interested parties on the ground. The Inmarsat ground segment consists of GES with antenna systems, Network Coordination Stations (NCS), Network Control Center (NCC), Satellite Control Center (SCC), and many GES terminals. The Iridium ground network comprises System Control Segment (SCS), Operational Support Network (OSN), and Network Control Station (NCS) and because of Inter-Satellite Links Iridium has only two gateway terminals.

ALSO, the ground segment constitutes the ground-based infrastructure necessary to support the operation of our satellites.

multimission ground network is made up of a central mission control, primary and backup ground stations for satellite control and data acquisition, and a distributed application ground segment for the generation of processed data and products.

3.4.1 CENTRAL FACILITY:

The Central Facility is responsible for the generation of level 1 processed satellite data and the generation of higher level 2 products.

• LEVEL 1 SATELLITE DATA

Raw sensor data are received from the ground stations and forwarded to dedicated processing facilities within the Central Facility. These facilities generate the level 1 data and products.

In the case of Meteo-sat image processing, line-by-line processing ensures that imperfections are removed during the generation of the level 1.5 satellite data. In particular, the data from the various on-board sensors are realigned by resampling, in order to make the image from each set of detectors coincide with the other images.

At the same time, the sampling removes the slight perturbations caused by the movement of the spacecraft, rectifying the image so it appears to come from the nominal location of the spacecraft (geometric correction). Adjustments to the individual data values are made according to calibration information. Once each line of the level 1.5 products is complete it is passed to the dissemination computers for immediate relay to users and to higher processing facilities within the Central Facility and the SAFs.

• LEVEL 2 PROCESSED PRODUCTS

In the case of Meteo-sat processing, the facility receives near-real-time level 1.5 data from the image processing system, which has performed a geometrical correction and removed imperfections in the images. A radiative transfer model is applied to compute expected radiances at the top of the atmosphere, as well as atmospheric correction tables.

Scenes analysis provides information on the surface type in each cloud-free image pixel. This output is then presented to the various meteorological product applications as a classified pixel map and segmented clustered scenes. Meteorological products are then generated.

In addition to the EUMETSAT satellite data, input from sources external to EUMETSAT is required for the processing and verification of some products. Independent meteorological observations are used to verify a subset of the products, and verification results are stored with the products.

● PRODUCT VALIDATION

Products generated within the Central Facility are subject to a validation processes to ensure the correct quality of product has been reached. All products need to meet validation criteria before an 'operational' product validation status can be applied. Corresponding validation reports are made available at the time of the product release.

ARCHIVING DATA AND PRODUCTS

All data and products generated within the Central Facility are archived in the EUMETSAT Data Centre and can be retrieved, on demand, by users.

3.4.1.1 MISSION CONTROL:

● Mission Control Centers (MCC), based at our headquarters are responsible for the safe operations of all satellites.

Comprised of two main control rooms, one for the geostationary missions and the other for the low Earth orbit missions, it provides the necessary monitoring and control of all operational satellites and the associated ground infrastructure.

It also ensures the continued communication with, and data acquisition from, satellites though a network of primary and back-up ground stations. Teams of satellite and ground segment controllers work around the clock, supported by teams of on-call operators and maintenance engineers.

(Fig. 3.4) Space-Craft Commanding and Control[16]

3.5 Types of Ground Systems

- Telemetry, Tracking and Control Stations (TT&C).
- Communications Stations.

3.5.1 Telemetry, Tracking and Control Stations (TT&C)

The telemetry, tracking, and control (TT&C) subsystem of a satellite provides a connection between the satellite itself and the facilities on the ground. The purpose of the TT&C function is to ensure the satellite performs correctly. As part of the spacecraft bus, the TT&C subsystem is required for all satellites regardless of the application.

We describe the three major tasks that the TT&C subsystem performs to ensure the successful operation of an applications satellite:

(1) the monitoring of the health and status of the satellite through the collection, processing, and transmission of data from the various spacecraft subsystems.

(2) the determination of the satellite's exact location through the reception, processing, and transmitting of ranging signals.

(3) the proper control of satellite through the reception, processing, and implementation of commands transmitted from the ground. Some advanced spacecraft designs have evolved toward "autonomous operations" so that many of the control functions have been automated and thus do not require ground intervention except under emergency conditions.

Onboard each satellite, the connection between the spacecraft and the command segment is achieved by the Telemetry, Tracking, and Control (TT&C) subsystem. As can be deduced from its name, this subsystem has three specific tasks that must be performed to ensure the ability of the satellite to successfully achieve any application:

1. Telemetry: The collection of information on the health and status of the entire satellite and its subsystems and the transmission of this data to the command segment on the ground. This requires not only a telemetry system on the spacecraft but also for a global network of ground stations around the world to collect the data, unless, of course, the application satellite network includes inter-satellite links that are capable of relaying the data to a central collection point.

2. Tracking: The act of locating and following the satellites to allow the command segment to know where the satellite is and where it is going, again this requires a ranging system on the spacecraft and a data collection network on the ground that allows this ranging and tracking function to work.

3. Control: The reception and processing of commands to allow the continuing operation of the satellite in order to provide the service of interest, again a ground system is required.

These tasks must be performed for both of the major components of the satellite: the payload and the spacecraft bus.

3.5.1.1 Telemetry: Providing Health and Status Updates for

The Satellite:

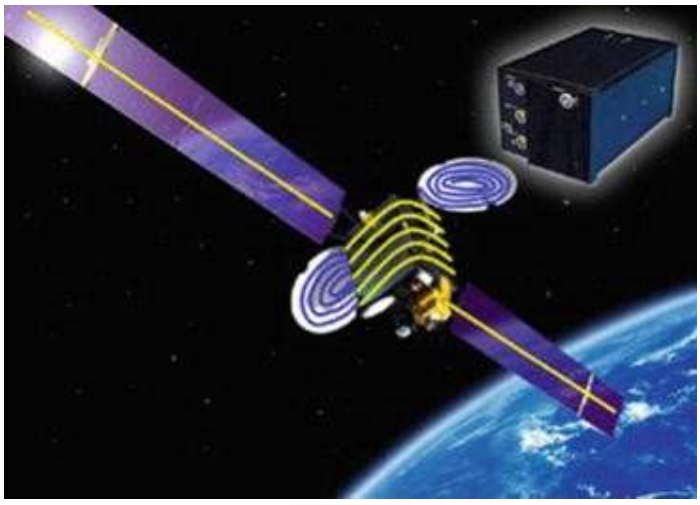
It is the task of the command segment located on the ground to provide the commands that will keep the satellite operating as required. However, before any commands can be issued or even chosen, the team on the ground must know what the status of the satellite is at any given time. Telemetry is the collection of measurements and onboard instrument readings required to deduce the health and status of all of the subsystems on the satellite. The TT&C subsystem must collect, process, and transmit this data from the satellite to the ground.

The first step in providing status updates to the ground is the collection of the measurements required by the command segment. Measurements related to the health and status of the satellite include:

- The status of resources (e.g., propellant supply and the health and charging status of batteries)
- The attitude of the satellite (e.g., the readings from sun and star trackers or RF tracking systems)
- The mode of operation for each subsystem (e.g., the on/off state of a heater)
- The health of each subsystem (e.g., output from the solar panels)

These measurements are not only necessary for the spacecraft bus, but also for assessing the health of the payload. On a communications satellite, the telemetry data would include information such as the switching configuration for the routing of signals, power output of the transponders, the direction the antenna is pointed in, or the health and status of imaging systems. All of these measurements are collected with various sensors such as thermometers, accelerometers, and transducers that provide outputs in such forms as measured resistance, capacitance, current, or voltage. The design of a spacecraft for the collection of data from such physical sensors

and the associated wiring required for gathering this information concerning the health and status of the spacecraft and payload can lead to a noticeable mass and cost penalty. For example, some larger communications or remote sensing satellites can have up to 500 temperature sensors onboard the spacecraft. The collection of data from such a large number of sensors can lead to an extensive wiring harness (500 temperature sensors × 2 wires = 1,000 wires). This has led to ongoing research related to new alternatives for collecting information such as the European Space Agency's Fiber Optic Satellite (FOSAT) project which uses fiber optics rather than conventional wiring in order to more efficiently gather data on the health of a satellite.[17]



(Fig.3.5)
Artist Depiction of FOSAT Using Fiber Optics for Health Monitoring (Courtesy of ESA)[17]

The use of these sensors to gather the required measurements is only the first step in providing telemetry from the satellite to the command team on the ground. The second step is the processing of the measurements. This processing includes the conversion of analog measurements to digital information as well as formatting of all the measurements for effective and, if required redundant, transmission to Earth. The processing of telemetry data involves two key factors. These two factors involve the nature of the

automation patterns of the spacecraft and the data storage algorithms. These will typically be different for each particular application satellite network. These will be based on the specific applications and mission parameters for each application satellite system.

Automation refers specifically to the ability of the satellite to interpret and respond to the telemetry measurements without interaction with the command segment. This allows the satellite to issue commands to the subsystems directly versus receiving and processing the commands from the ground. Automation can typically be found in response to predicable or common faults in certain subsystems that require actions such as placing components on standby when they operate outside of a range of parameters. Automation allows instantaneous actions to be taken onboard of a satellite and the degree of autonomous operation can vary widely among various applications satellites and the sophistication of their software. Automation can lead to three particular requirements for the TT&C subsystem as noted below (Pisacane-2005):

1.First and foremost is the ability of the onboard software to properly identify telemetry that indicates a subsystem is acting incorrectly and the corresponding ability to identify and process the correct response.

2.Closely associated with the first degree of automation is the related ability for the telemetry and the command components to communicate with each other and pass information back and forth on a nearly instantaneous basis.

3.Finally it is important for the software to have a diagnostic capability that allows the telemetry system to determine if an abnormal reading is caused by another subsystem or by errors in the TT&C subsystem itself.

Data storage of telemetry may also be required as transmissions down to the ground may not be feasible at any given moment. It is extremely costly (prohibitively so) to establish enough ground facilities for a global satellite system deployed in low earth orbit (LEO) to have constant contact with the command team unless there are inter-satellite links onboard all satellites that allows the relate of telemetry data to one or two central data processing locations. Due to this difficulty the satellite software and onboard computers must be able to process and store the data from the sensors on board while waiting for a viable window of communications to open up. The instrument readings and measurements are then transmitted along with other pertinent information such as when each measurement was collected. The cumulative data and exactly when it was collected is essential information for the ground team's data analysis particularly in case of anomalies.

An alternative to data storage is the use of a constellation of satellites that can relay telemetry from any given satellite to specific locations on Earth. One example is NASA's Tracking and Data Relay Satellite System (TDRSS) that consists of nine on-orbit satellites that relay communications from any other LEO satellite to its ground segment known as the White Sands Complex. This system can also be used for providing uplinks necessary for issuing commands to a satellite. European and Japanese systems have developed similar capabilities.

The final step the TT&C subsystem must perform in providing telemetry from the satellite to the command or ground segment is transmitting the data to the Earth. The principles, concepts, and hardware used for transmitting this type of data such as processing, commutation, multiplexing and antenna, and transponder design are described in Chaps. 12, "Regulatory Process for Communications Satellite Frequency Allocations," 13, "Satellite Radio Communications Fundamentals and Link Budgets," 14, "Satellite Communications Modulation and Multiplexing," 15, "Satellite Transmission, Reception and On-Board Processing Signaling and Switching," 16, "Satellite Communications Antenna Concepts and Engineering," 17, "Satellite Antenna Systems

Design and Implementation Around the World," 18, "Satellite Earth Station Antenna Systems and System Design," 19, "Technical Challenges of Integration of Space and Terrestrial Systems," 32, "Digital Image Acquisition: Preprocessing and Data Reduction," and 40, "Overview of the Spacecraft Bus."

The communications system used for downlinking telemetry may be the same system as that used for communicating the payload data or it may be an independent system depending on the satellite's application. Typical frequencies for the telemetry system include: S-band (2.2–2.3 GHz), C-band (3.7–4.2 GHz), and Ku-band (11.7–12.2 GHz). Other frequency bands can also be employed for different types of application satellite systems. Telemetry communications tend to have a bit-error-rate of approximately 10^{-5}. Telemetry systems that utilize Ku-band frequencies need to make some allowance for rain attenuation in the design of the TT&C (Larson and Wertz 1999).

3.5.1.1.1 Tracking: Locating and Following the Satellite:

In order to communicate with a satellite, whether it is to receive telemetry or send commands, the command segment must be able to locate, range, and track a satellite accurately. These ranging functions are part of the task of tracking which is performed by the TT&C subsystem. The satellite must first be able to locate and lock onto transmissions between the ground station and satellite. Once the satellite is locked on, the TT&C subsystem determines the range, or line-of-sight distance between the satellite and the radial velocity of the satellite. This allows the command segment to know where the satellite is and where it is going.

The process of locating and locking onto a satellite from a ground station is known as carrier tracking. This is most commonly accomplished in application satellite operations by using a principle known as phase coherence. This involves creating what is called a "two-way-coherent." The

typical operational mode in this respect involves establishing the downlink communication frequency at a predetermined ratio of the uplink frequency. This allows a synchronization of their phases. Initially the satellite searches and validates a connection to the uplink frequency based on the predetermined parameters. The TT&C subsystem then implements commands to set the frequency of the downlink communications so it is related to the uplink frequency through a prespecified ratio. This not only allows a ground station to lock onto a satellite, but it allows it to do so quickly as the expected downlink frequency is already known. There are standards for this process. For instance, the transponders tracking setting for the NASA Ground Spaceflight Tracking and Data Network (GSTDN) which sets the downlink/uplink ratios 240/221.

Determining the range between a satellite and a ground station is typically achieved through the use of tones or pseudo-code. The tone or code is modulated to the uplink frequency and when the satellite recognizes it, the TT&C subsystem adds the same tone or code to the downlink. The command segment can then calculate the round-trip time required for that tone and use that information to calculate the distance between the ground station and satellite. With the range (i.e., the distance to the spacecraft) established, the actual location of the satellite can be determined by using the pointing information of the satellite to determine the satellite's azimuth and elevation angles.

An alternative means for determining the range also allows for the radial velocity of the satellite to be determined. This method uses the Doppler shift of the frequencies of the uplink and downlink to determine the satellite's location and velocity. As discussed in earlier chapters, the Doppler effect is the change in frequency of the transmissions caused by the relative movement between the transmitter and the receiver. When the satellite is approaching a ground station, the frequency it receives is higher that the frequency transmitted. When the satellite is moving away from the ground station, the frequency it receives is lower than the frequency transmitted. This is also true for the frequency of the transmissions going from the satellite to the ground station. One issue with using the Doppler

effect to determine the location of a satellite is that there are always two locations, the true or nominal location and the virtual or mirror location that are possible at any single point in time. In order to account for this the TT&C subsystem has to apply processing algorithms to determine which location is correct. Satellites, such as Argos, use two positioning algorithms: least squares analysis and Kalman filtering. The Doppler shift method is best applied to satellites in relatively low orbits.

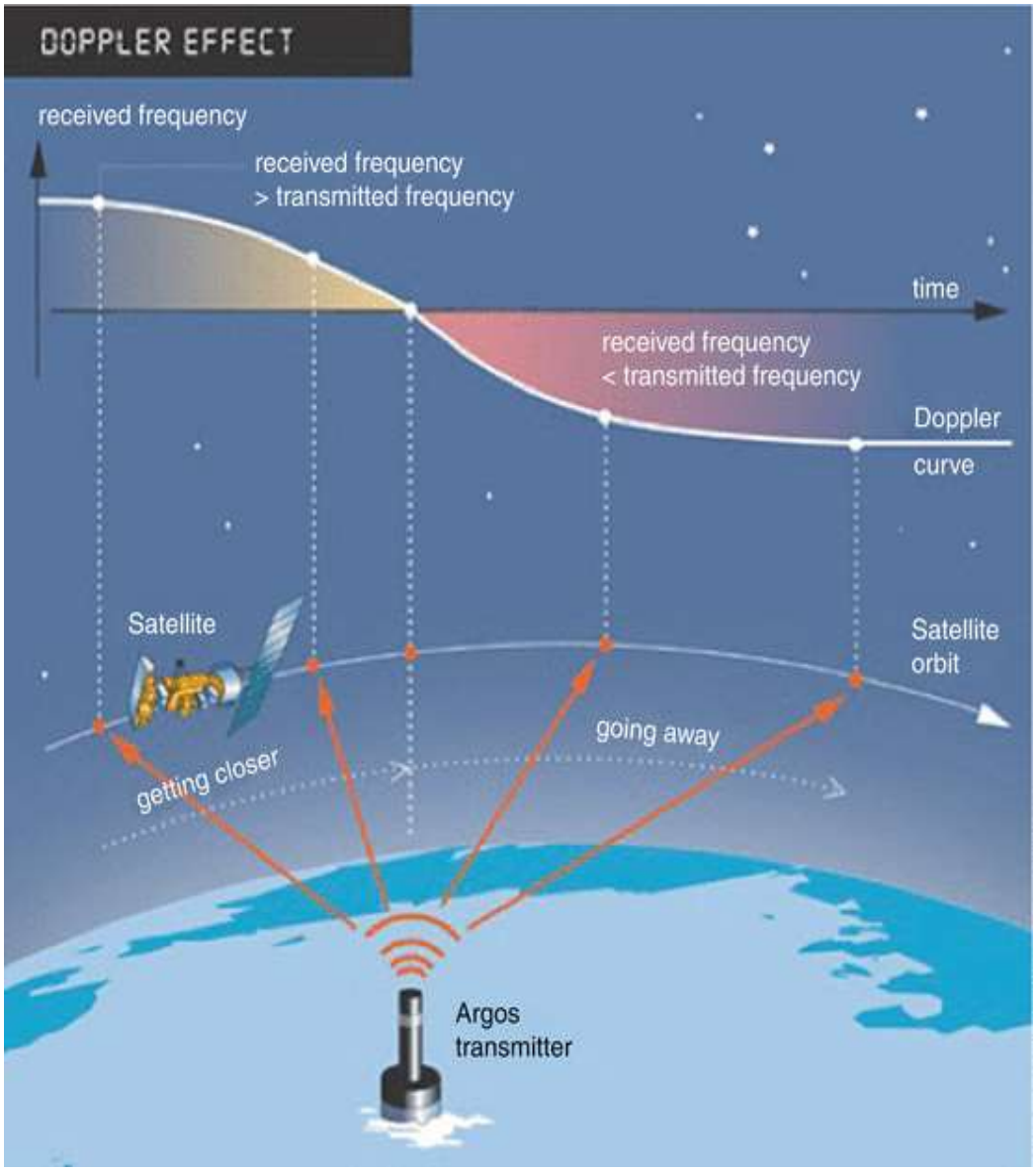


Fig. 3.6

Using the Doppler effect for tracking a satellite[17]

3.5.1.1.2 Control: Commanding the Spacecraft Bus and Payload Of the Satellite:

Control or command is the third task of the TT&C subsystem and it is the act of ensuring the satellite's spacecraft bus and payload do what is necessary to meet the objectives of its particular mission. Allowing control of the satellite requires that the TT&C subsystem receive, process, and implement the commands required by the command segment on the ground. As discussed briefly earlier, some commands may be automated through the use of onboard software that implements predefined commands upon recognition of specific conditions. Some satellites designed for "autonomous operation" carry this degree of automation to very sophisticated levels.

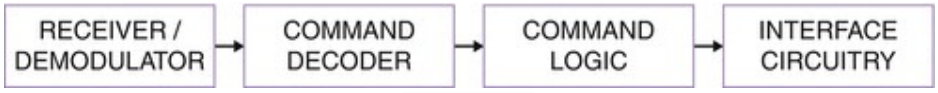
Commands are used to reconfigure a satellite or its subsystems to respond to mission conditions. Commands may include switching subsystems and components between on and off states or changing the operating mode in another manner. The commands may be used to control the spacecraft guidance and attitude control or deploy structures such as solar arrays or antennas. Finally, commands may come in the form of software programs that are to be uploaded into the onboard computer to control components on an ongoing basis.

The first step of the command system is to receive the data from the ground through its communication system. This system uses the same principles and technologies as described in Chaps. 12, "Regulatory Process for Communications Satellite Frequency Allocations," 13, "Satellite Radio Communications Fundamentals and Link Budgets," 14, "Satellite Communications Modulation and Multiplexing," 15, "Satellite Transmission, Reception and On-Board Processing Signaling and Switching," 16, "Satellite Communications Antenna Concepts and Engineering," 17, "Satellite Antenna Systems Design and Implementation Around the World," 18, "Satellite Earth Station Antenna Systems and

System Design," and 19, "Technical Challenges of Integration of Space and Terrestrial Systems." Typical frequencies for the command system include: S-band (2.2–2.3 GHz), C-band (3.7–4.2 GHz), and Ku-band (11.7–12.2 GHz). Control communications tend to have a bit-error-rate of approximately 10^{-6} (Keesee-2003).

This rate is an order of magnitude less than that noted for the telemetry communications due to the importance of ensuring that the commands issued by the ground are recognized correctly by the TT&C subsystem. Typical data rates required for command systems range from 500 to 1,000 kb/s.

Once the satellite has received and demodulated the uplink command transmissions (or a command is produced by the onboard computer), the command system includes three additional segments: the command decoder, the command logic, and the interface circuitry. The decoder reproduces the command messages and produces the lock/enable and clock signals required. The command logic validates the command and rejects it if there is any uncertainty regarding its authenticity. This logic then gets implemented through the interface circuitry that connects to the other systems on the satellite. Some application satellite systems have sophisticated security processes in place. These might require that at least some ground commands be authenticated from another location in order to be implemented. There have been instances where spurious commands (whether intentional or unintentional) have disabled application satellite networks.



(Fig. 3.7)
Command System Block Diagram

The command decoder collects and processes all incoming commands from sources such as the ground and the onboard computer. The decoder

includes an arbitration scheme that determines how each command is given priority in the processing queue. 4 Due to the criticality of the uplink commands, they are often encrypted and as noted above might also require authentication from another TT&C ground facility. Typical command messages include input checkerboard bits, synchronization bits, command bits, and error detection bits. The command itself includes the spacecraft's address and the command type. In virtually all operational application satellite systems both elements of the command must be verified. Command types include commands to flip a relay in a system, to pulse a piece of electronics, to change the output level of a component, or to request or send data to or from a component.

In some cases, the command may involve a whole sequence of events. In the case of the deployment of the Light Squared Land Mobile Satellite that was deployed in 2011, the 20 m antenna system did not normally deploy. In this case, a series of commands were sent to fire thruster jets in a sequence to "joggle" the spacecraft so that the antenna finally deployed.[17]

The command logic in the TT&C subsystem must verify and validate the command. This includes ensuring that the commands are being sent to the correct spacecraft or that the command itself is valid. Additionally, the timing of the command must be valid and the command itself must be authenticated. Once the logic is used to process the command, the TT&C subsystem activates the interface circuitry as necessary depending on the type of command. In the case of trouble shooting or failure-recovery operations, there may be a need to override constraints in the onboard software to allow higher risk commands to be executed.

3.5.2TT&C System Design Aspects

Satellites must be designed to operate correctly for the entire life of their mission. The TT&C subsystem is a critical part of ensuring that the satellite performs as required and can react to changes in conditions at the satellite (either internal or external). Because of this, the system must go through stringent quality control and testing of its components before they are used on a satellite. Additionally, key portions of the subsystem are designed to have redundant components to ensure that if one part fails; another is still available for use. If the satellite is designed for a 15-year time for instance, the TT&C subsystem might even be designed with a mean time to failure of 18–20 years.

One of the major trade-offs in designing a TT&C system is how complex the system must be to meet the goals of the mission. The more complex the system, it will be able to provide more telemetry and process more commands. The disadvantage is that this complexity typically leads to more components and therefore more mass and cost to the overall spacecraft. Most of the complexity of a TT&C subsystem is actually contained in the software that includes diagnostics to determine if a particular pathway in an onboard switch is somehow defective. The good news is that one can often update or reengineer software so that improved software can be uploaded after launch.

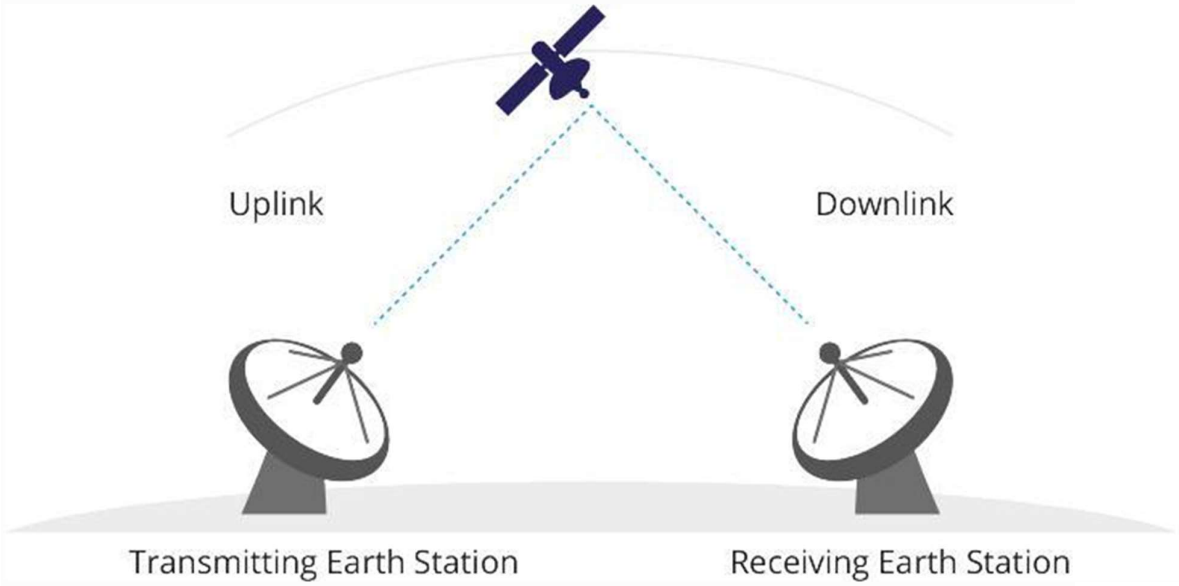
Connected to the concept of complexity, it the aforementioned use of automation. A designed must look at the requirements for the mission and determine how much telemetry and command should be dealt with onboard the satellite and how much should flow through the command segment on the ground. A thorough review of how the processing scheme for each command impacts the overall mission reliability and overall cost should be performed prior to launch and reviewed periodically to see if new software

could help insure improved and more reliable performance. In some cases, it may be more cost-effective to deal with situations through onboard computing, while other situations may be more cost-effectively dealt with through analysis on the ground.

3.5.3 Transmitting and Receiving in satellite

- Data is transmitted in units (frames).
- Each frame has a certain length of data bits, together with error detection or correction bits. Additional information such as acknowledgements may be included in a control field. Frame header and trailer identify frame start and end.
- Data in a frame may be arranged in layers. For example a long frame containing data from different sources (e.g. temperature and current) may include an identification for the data type, error correction bits, a header and a trailer for each source.
- The long frame containing these frames may be segmented into shorter frames of fixed length including check bits, headers and trailers, etc.

- The whole frame containing the segmented frames may include error correction, control field, a header and a trailer.



(Fig. 3.8) Transmitting and Receiving[18]

3.5.3.1Tracking

- Angle tracking of satellite by the ground station antenna.
- with a narrow beam.
- Range determination.
- Range rate determination (using Doppler shift).
- Frequency tracking.

3.5.3.2Ranging

Determines satellite distance, required, together with satellite angles, for determining satellite trajectory accurately.

Based on radar principle:

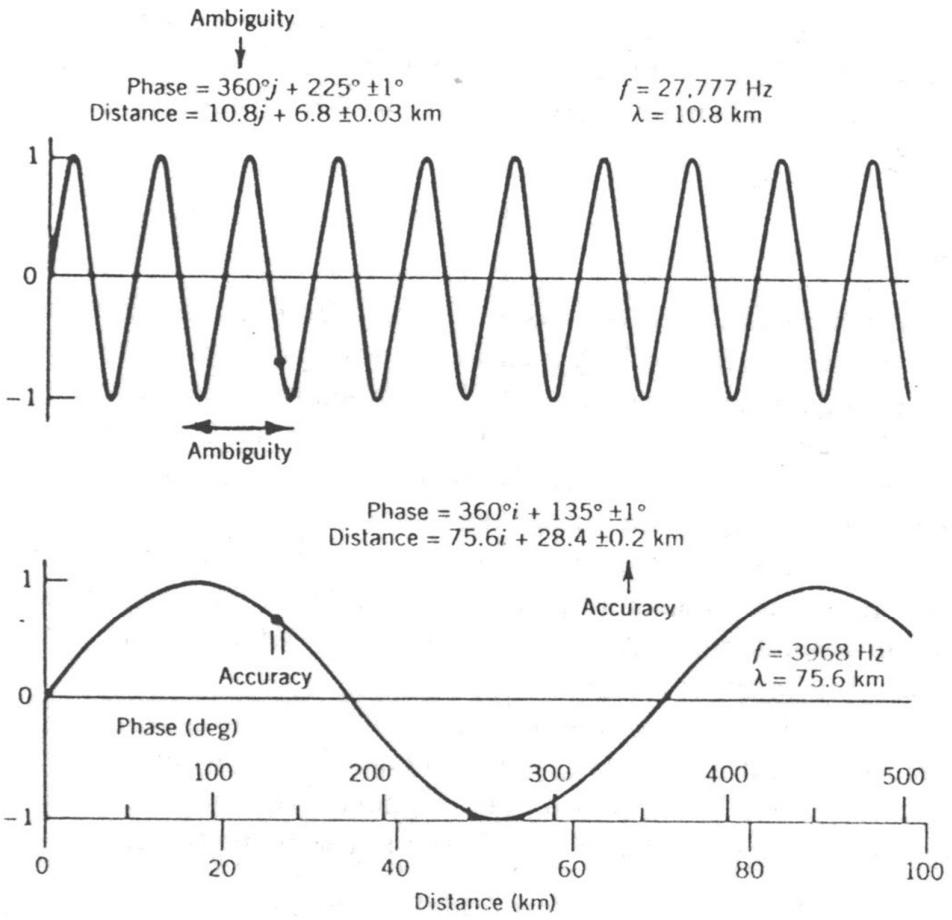
A signal is transmitted to the satellite. The satellite coherently retransmits the signal to the earth station. The path delay determines the distance.[18]

Methods:

- Multi-tones.
- Pseudo random signal.

3.5.3.3Multi-tones Ranging

Low frequency tones modulated on a carrier.The range is determined by the difference between the phases of the signal transmitted from the ground station and the signal returned by the satellite, together with the wavelength of the tone.If the tone wavelength is longer than twice the range, the phase determines the range directly. With 700 km range, the frequency is nearly 300000/1400 = 200 HZ.



(Fig. 3.9) Frequency Ranging[18]

- An accuracy of 1° in phase corresponds to distance accuracy of 1/360 of wavelength, which is a large distance (about 4 km).

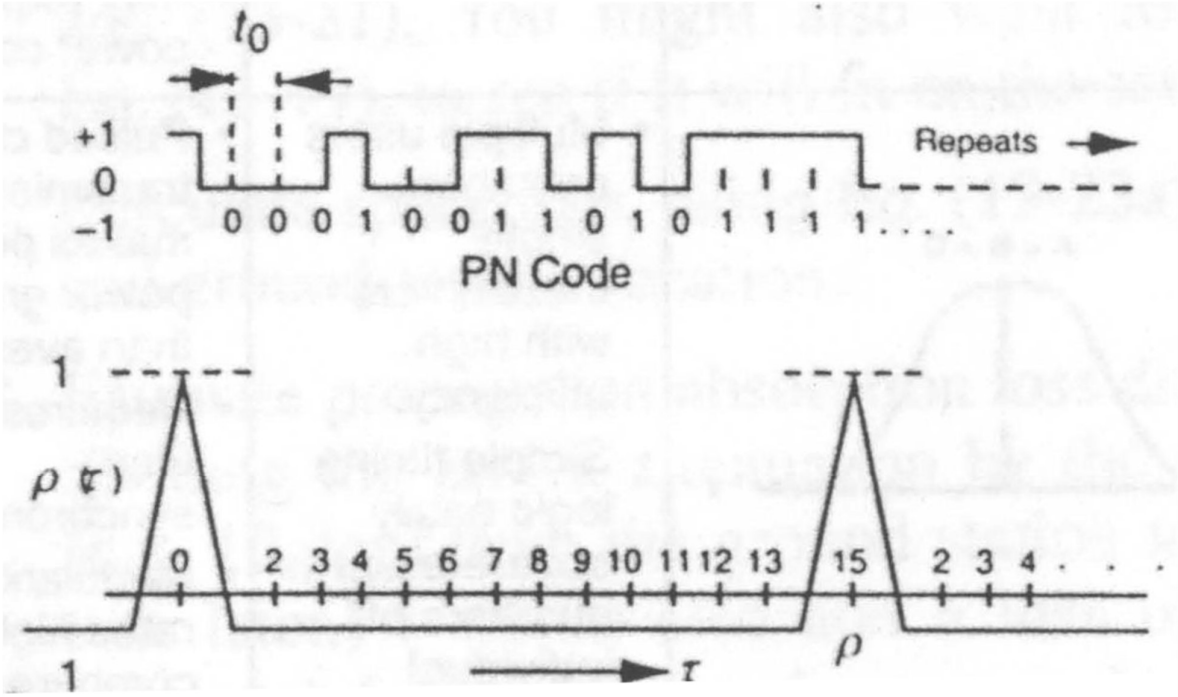
- For better distance accuracy use a shorter wavelength. The distance estimation becomes ambiguous by multiples of the shorter wavelength.

- The use of multi-tones resolves the ambiguities and attains high accuracy.

3.5.3.4Pseudo-Random signal ranging

- A long pseudo-random stream (pattern) of bits (chips) is transmitted (modulated on the carrier frequency) by the ground station, and the stream retuned from the satellite is compared (correlated) with the transmitted stream to determine the delay.

- Correlation is done by multiplying each of the bits of the received stream with the bits of the transmitted stream and adding the products. When the two streams coincide the result of correlation is maximum (all products become positive).



(Fig. 3.10) Pseudo-Random Signal Ranging[18]

- For no ambiguity in range determination, the pattern length should be longer than twice the range distance.

- With a large number of bits (chips) in the transmitted pattern, each chip corresponds to a short distance, thus the correlation to within a fraction of a chip leads to high accuracy in range determination.

- Misrsat-1 uses PR signal ranging.

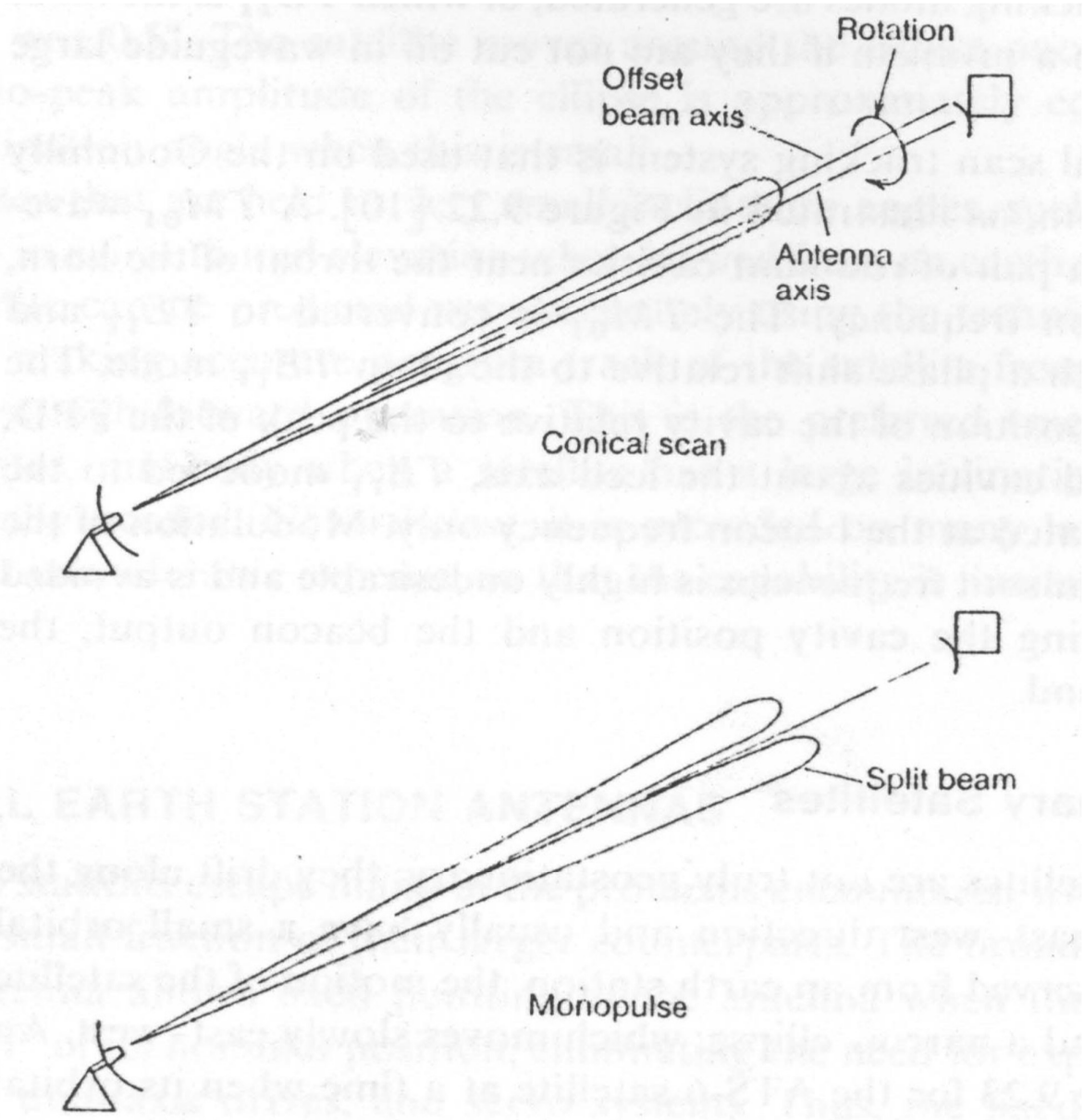
- The chip rate is about 500 kb/s (Chip length = 600m) leading to distance accuracy of about 50 m.

- The sequence length is about 26000 chips (chip length = 0.6 km) corresponding to an unambiguous distance of about 7000 km.

3.5.3.5Angle tracking

Angle determination of the satellite is mainly based on the known orbit. Once the ground station antenna acquires the satellite, the antenna tracks the satellite. The narrow beam of the satellite and the tracking system determine the satellite angles accurately. These angles are also used to update the satellite trajectory.[18]

❑ The ground station antenna tracks the satellite automatically by:
- Monopulse method.
- Conical scan method.



(Fig. 3.11): Methods Of tracking[18]

3.5.3.6Monopulse Tracking

To track the satellite angle in a certain direction (e.g. elevation angle), a special antenna is used whose radiation pattern has a null (no reception) on the antenna axis (in the direction of the satellite). As the satellite moves to a new elevation angle, the antenna begins to receive a signal from the satellite whose amplitude is proportional to the angular deviation. This signal is used as an error signal in a feedback loop to re-orient the antenna such that the antenna pattern null becomes oriented to the new satellite elevation angle.

- ➢ A similar antenna can be used to track the horizon angle of the satellite. The error signal is now generated if the horizontal angle of the satellite changes.

- ➢ This method is commonly used in tracking. Its basic use is in tracking radars.

3.5.3.7Conical scan tracking

In conical scan tracking, the narrow antenna beam rotates in a circle just surrounding the satellite. Its beam is thus moving (scanning) on the surface of a cone.

When the satellite moves in a certain direction, the satellite will lie in the antenna beam at a certain location of the beam during scanning leading to stronger received signal. This location determines the satellite direction, and a control loop automatically orients the antenna such that the circular footprint of the scanning beam just surrounds the satellite in its new direction.

3.5.3.8Frequency Acquisition and Tracking

- ➢ The satellite and the ground station need to lock their frequencies to each other exactly (for example to perform coherent demodulation). The oscillators have a drift, say of the order of 10^{-5}.

- ➢ Frequency search should be performed to lock the receiver local oscillator to the received signal (both in the satellite and the ground station).

- ➢ To speed the lock of the ground station local oscillator frequency to the frequency received from the satellite, the down link frequency is locked to the uplink frequency by a certain ratio, thus the ground station knows accurately the down link frequency which is derived from its uplink frequency.

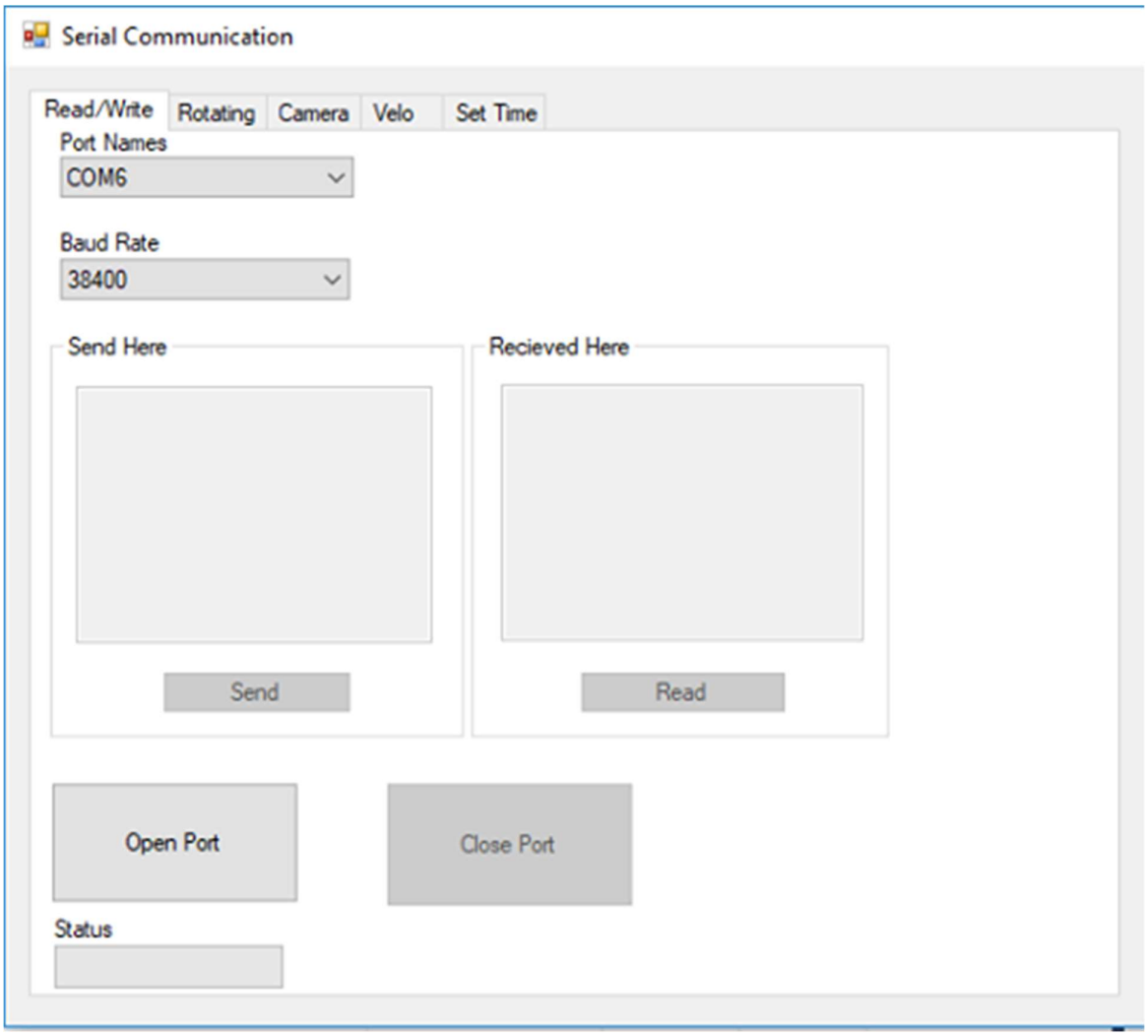
- ➢ In S-band the ratio of the down link frequency to the uplink frequency is 240/221, which is the ratio of any corresponding frequencies (of 2 MHz bandwidth) within the allowed uplink and down link frequency bands (2025 – 2110 MHz uplink and 2200 – 2290 MHz downlink).

- ➢ Frequency acquisition is first performed followed by frequency tracking using a phase locked loop.[18]

3.5.4Ground Station Software

The simulation of the Ground Station is not easy because it consists of a lot of equipment, so half of our team work on it by using C# language and visual studio to make software application that simulate the Ground Station.

Communication Session Configuration and Control



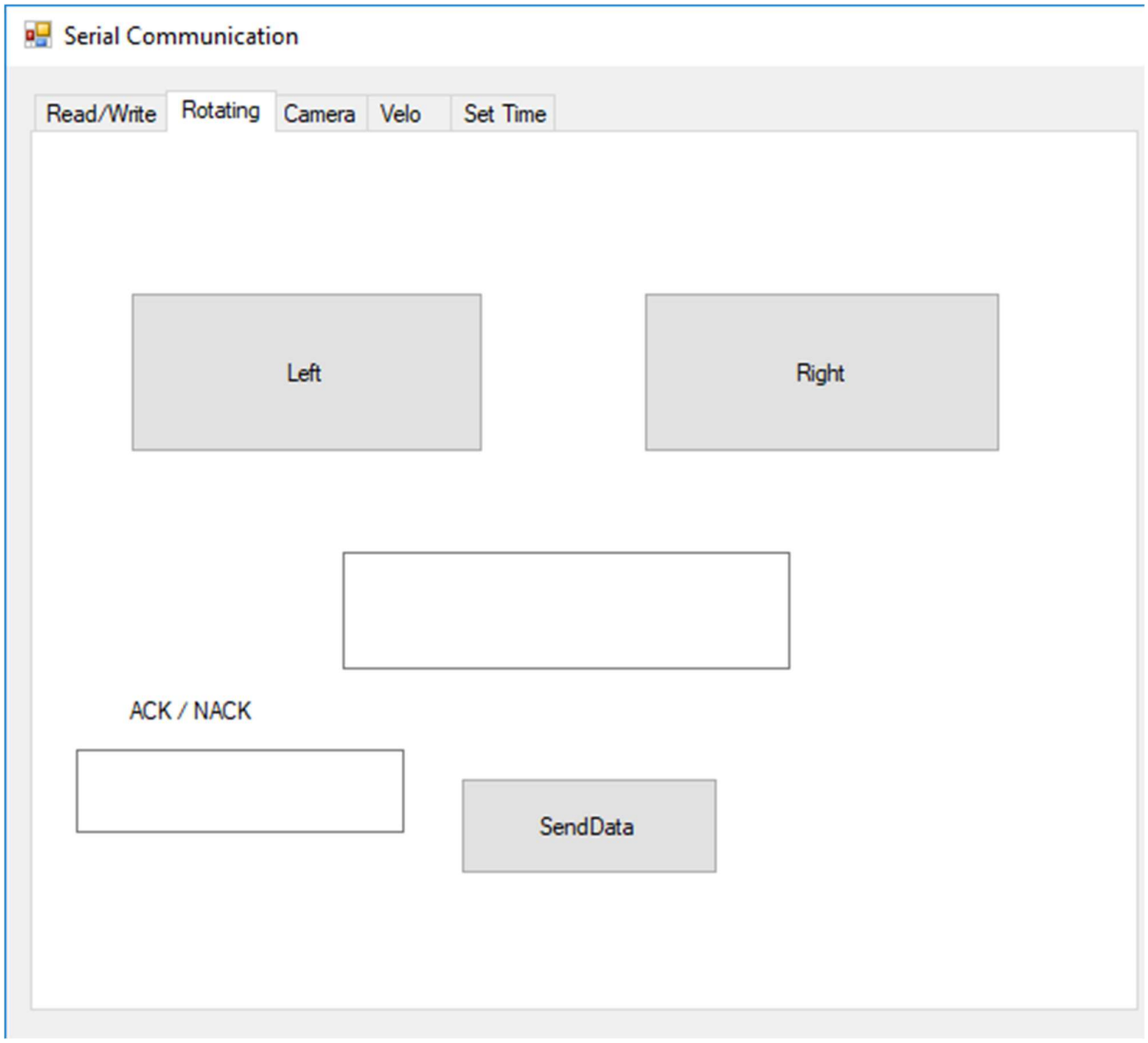
```
void getAvailablePorts()
{
    String[] ports = SerialPort.GetPortNames();
    comboBox1.Items.AddRange(ports);
}

private void button3_Click(object sender, EventArgs e)
{
    try
    {
        if (comboBox1.Text == "" || comboBox2.Text == "")
        {
            textBox2.Text = "Please select port settings";
        }
        else
        {
            serialPort1.PortName = comboBox1.Text;
            serialPort1.BaudRate = Convert.ToInt32(comboBox2.Text);
            serialPort1.Open();
            progressBar1.Value = 100;
            button1.Enabled = true;
            button2.Enabled = true;
            textBox1.Enabled = true;
            button3.Enabled = false;
            button4.Enabled = true;
        }
    }
    catch (UnauthorizedAccessException)
    {
        textBox2.Text = "Unauthorized Access";
    }
}
```

```
private void button4_Click(object sender, EventArgs e)
{
    serialPort1.Close();
    progressBar1.Value = 0;
    button1.Enabled = false;
    button2.Enabled = false;
    button4.Enabled = false;
    button3.Enabled = true;
    textBox1.Enabled = false;
}

private void button1_Click(object sender, EventArgs e)
{
    serialPort1.WriteLine(textBox1.Text);
    textBox1.Text = "";
}

private void button2_Click(object sender, EventArgs e)
{
    try
    {
        textBox2.Text = serialPort1.ReadLine();
    }
    catch (TimeoutException)
    {
        textBox2.Text = "Timeout Exception";
    }

}
```

Cyclic Redundancy Check (CRC) Function Which Helps in Detecting Errors

```
byte getCRC(byte[] message, int length)
{
    byte j, crc = 0;
    byte CRC7_POLY = 0x91;
    UInt32 aa = 1, c;
    int i;

    for (i = 1; i < length; i++)
    {
        crc ^= message[i];

        for (j = 0; j < 8; j++)
        {
            c = Convert.ToUInt32(crc) & aa;
            if (c > 0)
                crc ^= CRC7_POLY;
            crc >>= 1;
        }
    }
    return crc;
}
```

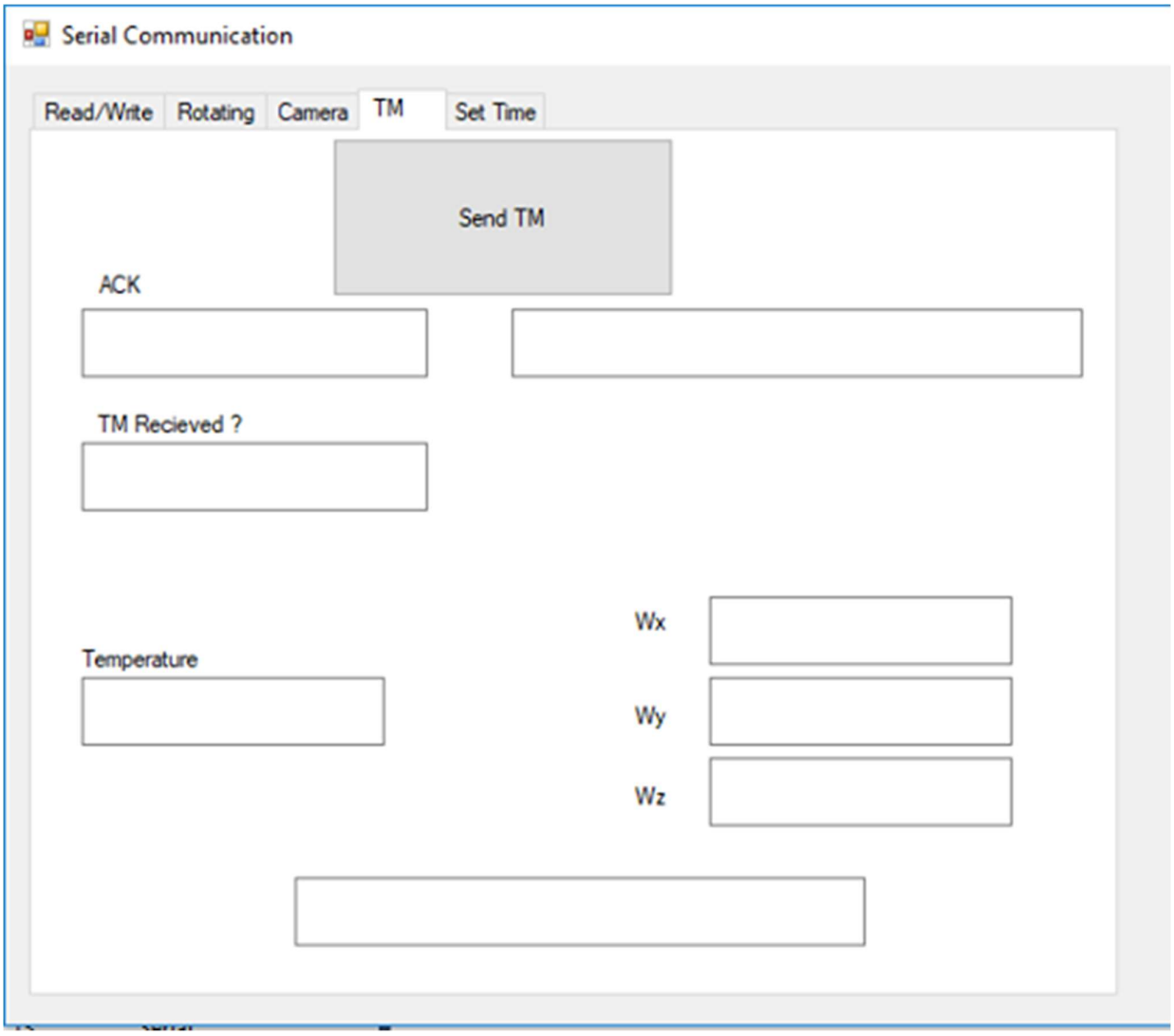
Controlling The Direction of The DC Motor and Detecting Errors By Means of Sending A Positive Or Negative Acknowledgement



```
private void button6_Click(object sender, EventArgs e)
{
    right[13] = getCRC(right, 13);
    textBox3.Text = BitConverter.ToString(right);
    serialPort1.Write(right, 0, 14);

    wait_3_bytes: recived_data_length = serialPort1.BytesToRead;

    if (recived_data_length >= command_respons_length)
    {
        // read recived data array >= 3 byte
        serialPort1.Read(recived_replay, 0, command_respons_length);
    }
    else
        goto wait_3_bytes;

    if (recived_replay[1] == 0xAC)
    {
        textBox7.BackColor = Color.Green;
        textBox7.ForeColor = Color.White;

        textBox7.Text = "satellite recive Command correctely ";

    }


    else
        if (recived_replay[1] == 0xA0)
        {
            textBox7.BackColor = Color.Red;
            textBox7.ForeColor = Color.White;

            textBox7.Text = "satellite recive wrong CRC; retransmt ";
        }
}
```

```
private void button5_Click(object sender, EventArgs e)
{
    //serialPort1.WriteLine(textBox3.Text);
   left[13] = getCRC(left, 13);
   textBox3.Text = BitConverter.ToString(left);
    //textBox3.Text = "";
   serialPort1.Write(left, 0, 14);

    wait_3_bytes: recived_data_length = serialPort1.BytesToRead;

    if (recived_data_length >= command_respons_length)
    {
        // read recived data array >= 3 byte
        serialPort1.Read(recived_replay, 0, command_respons_length);
    }
    else
        goto wait_3_bytes;

    if (recived_replay[1] == 0xAC)
    {
        textBox7.BackColor = Color.Green;
        textBox7.ForeColor = Color.White;

        textBox7.Text = "satellite recive Command correctely ";

    }
    else
        if (recived_replay[2] == 0xA0)
        {
            textBox7.BackColor = Color.Red;
            textBox7.ForeColor = Color.White;

            textBox7.Text = "satellite recive wrong CRC; retransmt ";
        }
}
```

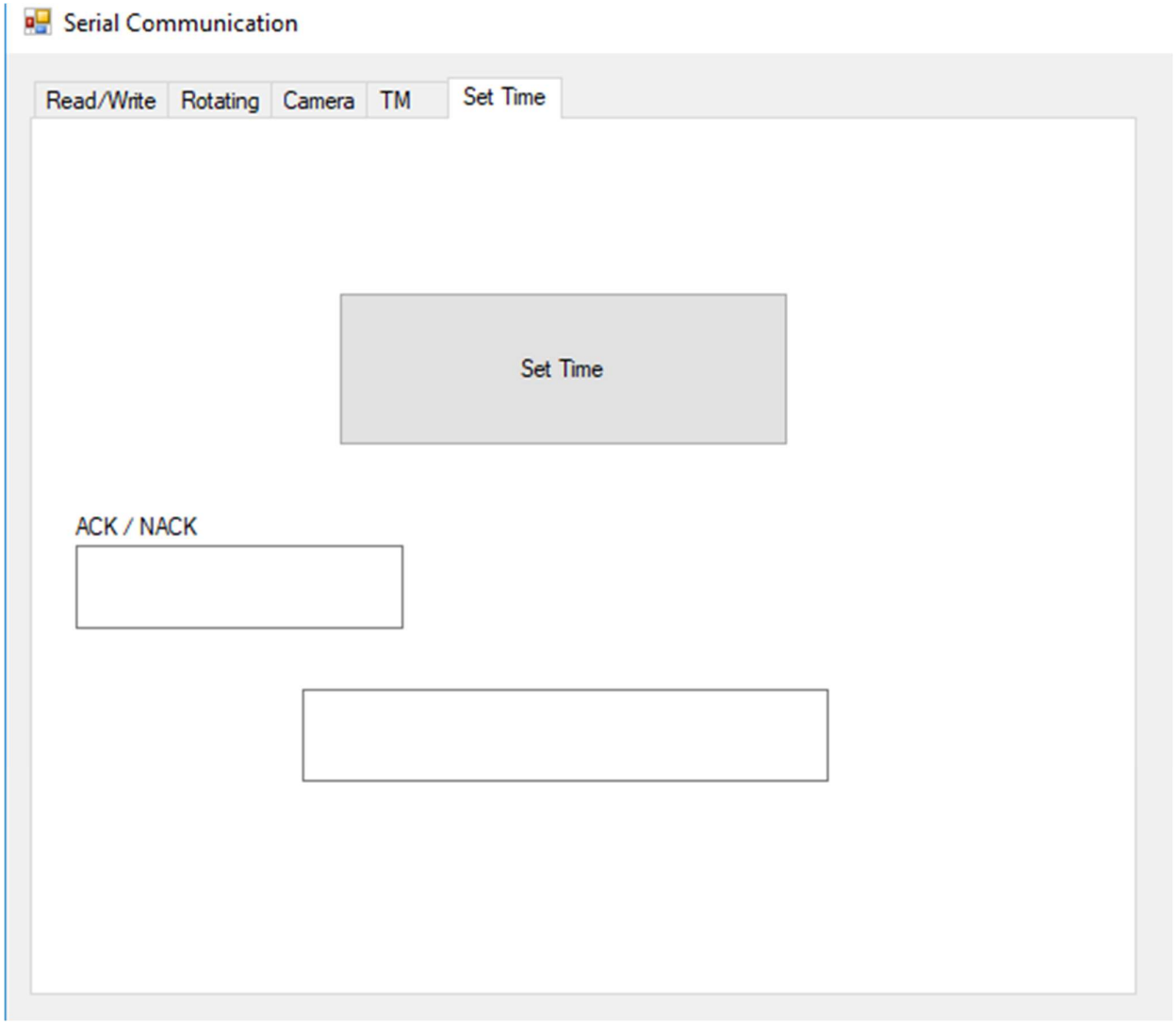
Telemetry Requesting and Receiving



```
private void button9_Click(object sender, EventArgs e)
{
    TM[13] = getCRC(TM, 13);
    textBox5.Text = BitConverter.ToString(TM);
    serialPort1.Write(TM, 0, 14);

   wait_3_bytes: recived_data_length = serialPort1.BytesToRead;

    if (recived_data_length >= command_respons_length)
    {
        // read recived data array >= 3 byte
        serialPort1.Read(recived_replay, 0, command_respons_length);
    }
    else
        goto wait_3_bytes;

    if (recived_replay[1] == 0xAC)
    {
        textBox8.BackColor = Color.Green;
        textBox8.ForeColor = Color.White;

        textBox8.Text = "satellite recive Command correctely ";
        command_recived_correctly = 1;
    }
    else
        if (recived_replay[2] == 0xA0)
        {
            textBox8.BackColor = Color.Red;
            textBox8.ForeColor = Color.White;
            textBox8.Text = "satellite recive wrong CRC; retransmt ";
        }
```

```
    if (command_recived_correctly == 1)
    {

    wait_21_bytes: recived_data_length = serialPort1.BytesToRead;
        if (recived_data_length >= Sat_TM_legth)

        {
            // read recived data array >= 9 byte
            serialPort1.Read(recived_TM, 0, Sat_TM_legth);
        }
        else

            goto wait_21_bytes;

//        if (recived_TM[20] == getCRC(recived_TM,20))
  //      {
            textBox13.BackColor = Color.Yellow;
            textBox13.ForeColor = Color.Black;
            textBox13.Text = "satelliet TM recived correctly";
            Temperature = BitConverter.ToSingle(recived_TM, 16);
            Wx = BitConverter.ToSingle(recived_TM, 8);
            Wy = BitConverter.ToSingle(recived_TM, 12);
            Wz = BitConverter.ToSingle(recived_TM, 4);

            textBox12.Text = Temperature.ToString();
            textBox9.Text = Wx.ToString();
            textBox10.Text = Wy.ToString();
            textBox11.Text = Wz.ToString();
            textBox14.Text = BitConverter.ToString(recived_TM);
     // }
    }

}
```

Time Synchronization Between Ground Station and The Satellite



```
private void button10_Click(object sender, EventArgs e)
{
    settime[13] = getCRC(settime, 13);

    textBox6.Text = BitConverter.ToString(settime);

    serialPort1.Write(settime, 0, 14);

wait_3_bytes: recived_data_length = serialPort1.BytesToRead;

    if (recived_data_length >= command_respons_length)
    {
        // read recived data array >= 3 byte
        serialPort1.Read(recived_replay, 0, command_respons_length);
    }
    else
        goto wait_3_bytes;

    if (recived_replay[1] == 0xAC)
    {
        textBox15.BackColor = Color.Green;
        textBox15.ForeColor = Color.White;

        textBox15.Text = "satellite recive Command correctely ";

    }
    else
        if (recived_replay[1] == 0xA0)
        {
            textBox15.BackColor = Color.Red;
            textBox15.ForeColor = Color.White;

            textBox15.Text = "satellite recive wrong CRC; retransmt ";
        }
}
```

4. Conclusion

The project includes a definition, declaration and design of CubeSat prototype including on board computer subsystem, ADCS subsystem, EPS , Payloads in addition to ground station simulator. The project guaranteed sending and receiving data between ground station and satellite with high accuracy due to the used error checking mechanism

At the end of the project we sure that we learned a lot of important knowledge such as programming and code writing, debugging and execution, prototype and model design and implementation, using microcontrollers, electronics and hardware interfaces and connections. However, the most important gain is the obtained learning skills specially that related to space engineering and satellite subsystems which there were not easy to be learned and understood without the gained experience by executing this project.

4.1 Future Work

We are proud of implementing this CubeSat Prototype which enables us to overcome the complexity of learning satellite subsystems and make it easy to be implemented.
There are some uncompleted issues because of the global

situation specially covid-19 restrictions such the camera that we want to import from America to use as Payload, and somethings need development like:

- Using magnetorquer as actuator.
- Use the magnetometer.
- Solar cell for battery charging.
- Use the solar tracker sensor.
- Antenna for communication
- Add thermal subsystem.
- Make the CubeSat prototype smaller in size.

References

[1] NASA Knows! for Students K-4

[2] Cubesat Project, Feasibility Case Stud ,Sara Mohammed Ali, University of Khartoum

[3] A Basic Guide to Nanosatellites - Alén Space

[4] Rani, B. S., Santhosh, R. R., Prabhu, L. S., Federick, M., Kumar, V., & Santhosh, S. (2010). A Survey to Select Microcontroller for Sathyabama Satellite's On Board Computer Subsystem.

[5] The avr microcontroller with assembly & C ,Mohamed Ali Mazidi ATmega32 8-bit microcontroller data sheet

[6] Embeded System Lectures, Kerelos Shenoda, Learn in Depth [7] ATmega32 8-bit microcontroller data sheet,Atmel.

[8] HC05 data sheet

[9] DS3231 RTC Module data sheet

[10] System Specifications for Attitude Determination and Control System, Project ECE3SAT.

[11] AlexSat, Ahmed farag (2011).

[12] MPU 6050 data sheet

[13] Bliley Technology

[14] SatSearch by Hywel Curtis

[15] Nn Guestational Aeronautics and space Administration.

[16] EUMETSAT

[17] Handbook of satellite Application by Arthur Norma Guest

[18] Introduction to Ground Control Station(GCS) design,Mostafa Yahia Mostafa,MISRSAT-2 Ground segment system engineer, Egyptian space program.

APPENDIX A

OBC Software

main.c file

```
#define F_CPU 16000000
#include <avr/interrupt.h>
#include "UARTt.h"
#include "delay.h"
#include "i2c.h"
#include "RTC.h"
#include "command.h"
#include "LCD.h"

u8 command_rec[14];

u8 telemetry[21]={0x00,0x01,0x00,0x10};

u8 ADCS_comm[7];

u8 ack[3]={0x00,0XAC,0x00};

u8 Nack[3]={0x00,0xA0,0x00};

float ADC_float,Vin;

volatile u8 x;

ISR(USART_RXC_vect)
{
	SerialReceive(command_rec,14);
	x++;
}


int main(void)
{
	//ADC_VoidInit();
	rtc_t rtc,rtc_read;
    u8 true_comm=0,false_comm=0;

	GPIO_voidInit();
	USART_init();
	ds3231_init();
	LCD_Init();
	GPIO_u8_WritePinDir(3,1,'c');
	GPIO_u8_WritePinDir(4,1,'c');

	DELAY_MS(100);

	/*Interrupt ENABLE*/
```

```
    sei();
       /* START all Subsystems (Turn oN RElAY 1) */
       //GPIO_u8_WritePinVal(4,1,'c');

//     ADC_VoidRead();

       while(1)
       {
              if (x==1)
              {
                     if(getCRC(command_rec,13)==command_rec[13])
                     {
                            /*The number of commands that delivered true*/
                            true_comm++;
                            /*i received true command */
                            SerialTransmit(ack,3);

                            /* command for ADCS  : ADCS ID = 0x0A */
              if(command_rec[0]==0x0A)
              {
                     ADCS_command(command_rec,ADCS_comm,telemetry);
              }
                        /* commands for OBC : OBC ID = 0x2A */
                      if ( (command_rec[0]==0x2A) && (command_rec[1]==0x0F) )
              {
                             RTC_SetDisplay(command_rec,&rtc,&rtc_read);
                      }
                  }

                  else
                  {
                         //ERROR

                         SerialTransmit(Nack,3);
                         false_comm++;
                         if (false_comm==3)
                         {
                                Emergency_mode();
                         }
                  }
                   x=0;
       }

       }
}

ISR(ADC_vect)
{
       u8 Low = ADCL;
       u16  TenBitResult = ADCH<<8 | Low ;
       ADC_float =TenBitResult;
       Vin = ADC_float*5/1024 ;

       if(Vin>4.5)
              GPIO_u8_WritePinVal(4,0,'c');
       else
           GPIO_u8_WritePinVal(4,1,'c');
```

```
	DELAY_MS(100);
	ADC_VoidRead();
}
```

ADCS Software

main.c File

```
#define F_CPU 8000000UL			/* Define CPU clock Frequency 8MHz */

#include <string.h>			/* Include string header file */
#include "LCD.h"		/* Include LCD header file */
#include "I2C.h"		/* Include I2C slave header file */
#include "motor.h"


#define Slave_Address				0x40
#define ACK				0xAC
#define NACK				0xA0

int main(void)
{
	char arr[7]={0};
	int8_t replay=0,y=0,x=0,count = 0;
	GPIO_voidInit();
	motor_init();
	move_forward();
	LCD_lcd_init();
	GPIO_u8_WritePortDir(0xff,'c');
	I2C_Slave_Init(Slave_Address);

	char telemetry_data[16];
	//float Xa,Ya,Za;
	char send_telemetry=0;
	Telemetry Xg,Yg,Zg,t;
	volatile int8_t Ack_status=0,telemetry=0,ack_ok=0;


	while (1)
	{
		if(I2C_Slave_Listen()==0)   /* Check for SLA+W or SLA+R */
		{
			do
			{
			  if (y!=0)
			     {
				arr[x]=count;
				x++;
```

```
                        if (getCRC(arr,6)==arr[6])
                        {
                               replay=ACK;
                           if(arr[2]==0x03)
                            {
                              set_speed_left(arr[5]);
                               move_forward();
                            }
                            else if(arr[2]==0x04)
                            {
                              set_speed_left(arr[5]);
                               move_backward();
                            }
                            else if(arr[2]==0x05)
                            {
                           telemetry=1;

                             }
                         }

                        else
                              replay=NACK;
                          }
                     y++;
                     count = I2C_Slave_Receive();/* Receive data byte*/
               } while (count != -1);/* Receive until STOP/REPEATED START */
                     count = 0;
                     x=0;
                     y=0;
          }
       else if (I2C_Slave_Listen()==1)
       {

              do
              {
                     if (send_telemetry==1)
                     {
                           static char i=0;
                            Ack_status = I2C_Slave_Transmit(telemetry_data[i]);
                            i++;
                           // _delay_us(2);
                            if (i==16)
                            {
                                   send_telemetry=0;
                                   i=0;
                            }
                     }

                    else
                   {
                      Ack_status = I2C_Slave_Transmit(replay);/* Send data byte
*/
                   }

              } while (Ack_status == 0);/* Send until Ack is receive */

                     if ( (telemetry==1))
```

```
                {
               ack_ok=1;
                }

  }

if ( (telemetry==1) && (ack_ok==1))
 {

                I2C_Master_Init();        /* Initialize I2C */
                telemetry=0,ack_ok=0;
                Gyro_Init();              /* Initialize Gyro */
                Read_RawValue();

               /* Divide raw value by sensitivity scale factor */
                Xa = Acc_x/16384.0;
                Ya = Acc_y/16384.0;
                Za = Acc_z/16384.0;

                Xg.mpu_data = Gyro_x/16.4;
                Yg.mpu_data = Gyro_y/16.4;
                Zg.mpu_data = Gyro_z/16.4;

                /* Convert temperature in /c using formula */
                t.mpu_data = (Temperature/340.00)+36.53;

               char i,j=0;
               for (i=0;i<16;i++)
               {
                            if ((0<= i)&&(i <4))
                            {
                                   telemetry_data[i]=Zg.mpu_data_byte[j];

                            }

                            else if ((4<= i) &&(i<8))
                           {
                               telemetry_data[i]=Xg.mpu_data_byte[j];

  }

       else if ((8<= i)&&(i <12))
  {
    telemetry_data[i]=Yg.mpu_data_byte[j];

  }

      else if ((12 <= i)&&(i<16) )
       {
              telemetry_data[i]=t.mpu_data_byte[j];

        }
              j++;
           if (j==4)
          {
              j=0;
            }
```

```
                    }
                     send_telemetry=1;
                     I2C_Slave_Init(Slave_Address);

        }


         }
}
```

APPENDIX B : Data Sheets

Atmega32 Data sheet

Features

• High-performance, Low-power Atmel®AVR® 8-bit Microcontroller
• Advanced RISC Architecture
– 131 Powerful Instructions – Most Single-clock Cycle Execution
– 32 × 8 General Purpose Working Registers
– Fully Static Operation
– Up to 16 MIPS Throughput at 16MHz
– On-chip 2-cycle Multiplier

• High Endurance Non-volatile Memory segments
– 32Kbytes of In-System Self-programmable Flash program memory
– 1024Bytes EEPROM
– 2Kbytes Internal SRAM
– Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
– Data retention: 20 years at 85°C/100 years at 25°C(1)
– Optional Boot Code Section with Independent Lock Bits
In-System Programming by On-chip Boot Program
True Read-While-Write Operation
– Programming Lock for Software Security

• JTAG (IEEE std. 1149.1 Compliant) Interface
– Boundary-scan Capabilities According to the JTAG Standard
– Extensive On-chip Debug Support
– Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface

• Peripheral Features
– Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
– One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
– Real Time Counter with Separate Oscillator
– Four PWM Channels
– 8-channel, 10-bit ADC
8 Single-ended Channels
7 Differential Channels in TQFP Package Only
2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
– Byte-oriented Two-wire Serial Interface
– Programmable Serial USART
– Master/Slave SPI Serial Interface
– Programmable Watchdog Timer with Separate On-chip Oscillator
– On-chip Analog Comparator

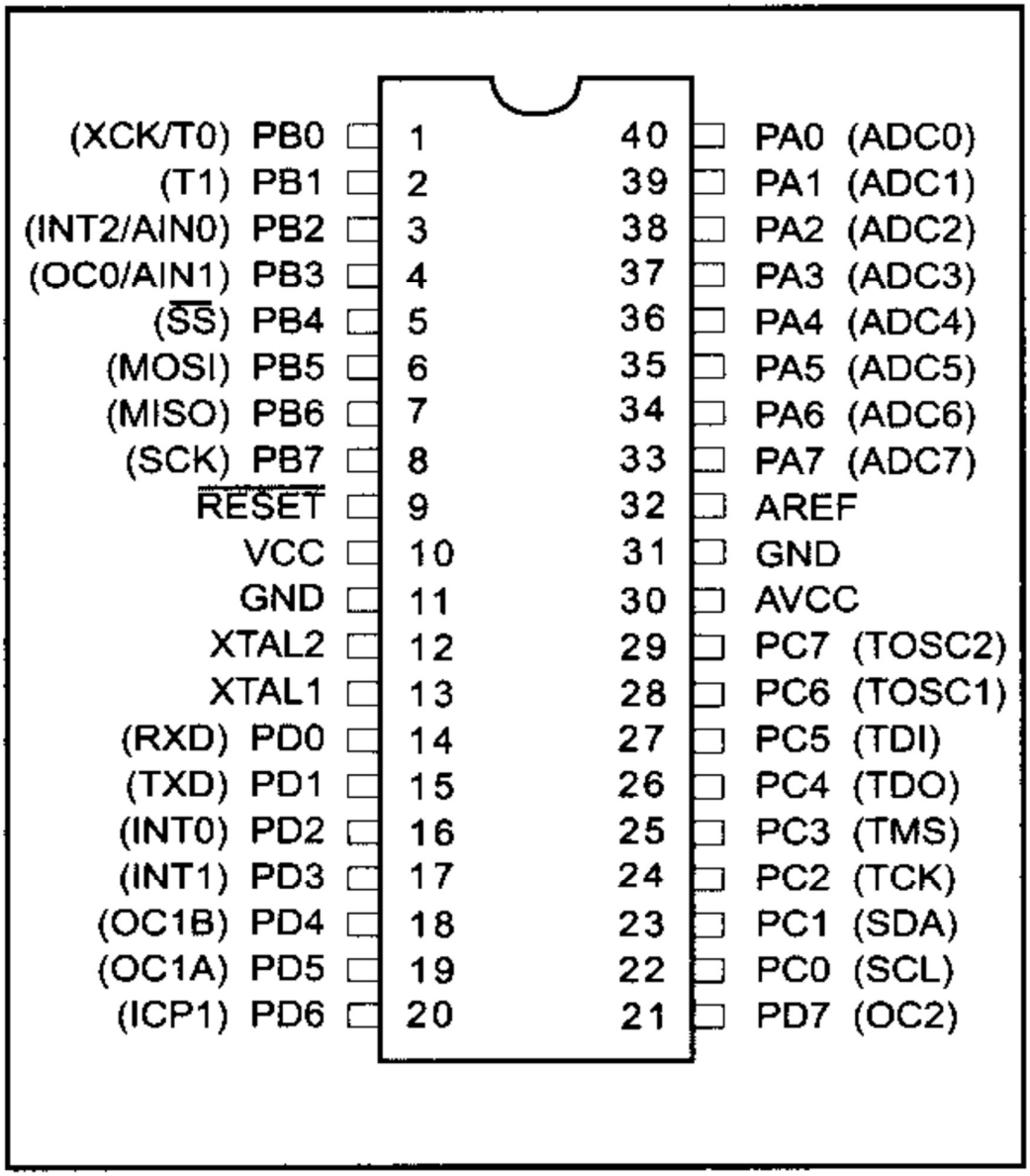
• Special Microcontroller Features
– Power-on Reset and Programmable Brown-out Detection
– Internal Calibrated RC Oscillator
– External and Internal Interrupt Sources
– Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby

• I/O and Packages
– 32 Programmable I/O Lines
– 40-pin PDIP, 44-lead TQFP, and 44-pad QFN/MLF

• Operating Voltages
– 2.7V - 5.5V for ATmega32L
– 4.5V - 5.5V for ATmega32

• Speed Grades
– 0 - 8MHz for ATmega32L
– 0 - 16MHz for ATmega32

• Power Consumption at 1MHz, 3V, 25°C
– Active: 1.1mA
– Idle Mode: 0.35mA
– Power-down Mode: < 1µA



HC05 Bluetooth module Data Sheet

Specifications

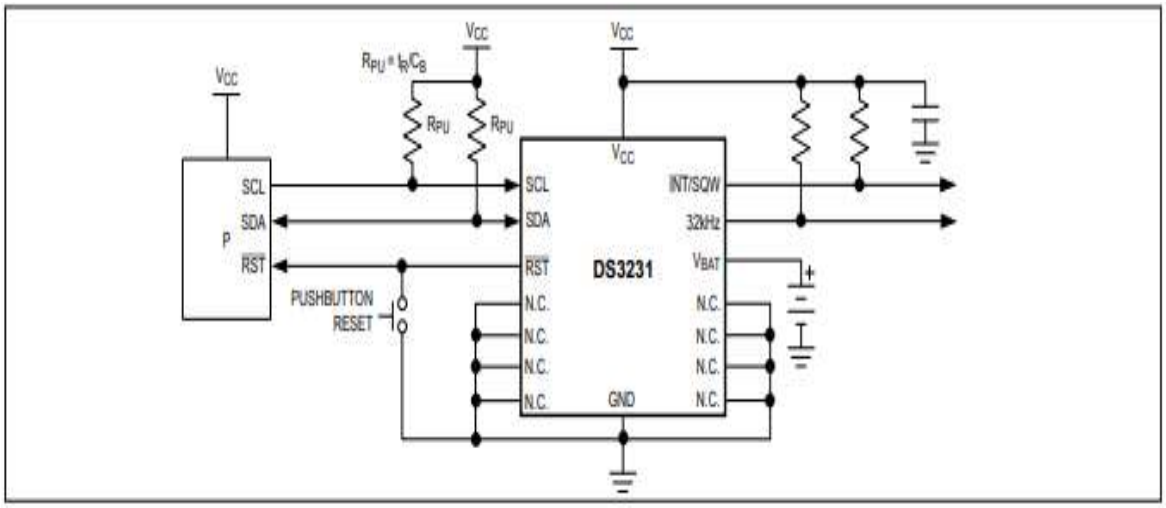
Hardware features

- Typical -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low Power 1.8V Operation ,1.8 to 3.6V I/O
- PIO control
- UART interface with programmable baud rate
- With integrated antenna
- With edge connector

Software features

- Default Baud rate: 38400, Data bits:8, Stop bit:1,Parity:No parity, Data control: has. Supported baud rate: 9600,19200,38400,57600,115200,230400,460800.
- Given a rising pulse in PIO0, device will be disconnected.
- Status instruction port PIO1: low-disconnected, high-connected;
- PIO10 and PIO11 can be connected to red and blue led separately. When master and slave are paired, red and blue led blinks 1time/2s in interval, while disconnected only blue led blinks 2times/s.
- Auto-connect to the last device on power as default.
- Permit pairing device to connect as default.
- Auto-pairing PINCODE:"0000" as default
- Auto-reconnect in 30 min when disconnected as a result of beyond the range of connection.

DS3231 RTC Module Data Sheet



- **Pin Description**

PIN	NAME	FUNCTION
1	32kHz	32kHz Output. This open-drain pin requires an external pullup resistor. When enabled, the output operates on either power supply. It may be left open if not used.
2	V_{CC}	DC Power Pin for Primary Power Supply. This pin should be decoupled using a 0.1µF to 1.0µF capacitor. If not used, connect to ground.
3	$\overline{INT}$/SQW	Active-Low Interrupt or Square-Wave Output. This open-drain pin requires an external pullup resistor connected to a supply at 5.5V or less. This multifunction pin is determined by the state of the INTCN bit in the Control Register (0Eh). When INTCN is set to logic 0, this pin outputs a square wave and its frequency is determined by RS2 and RS1 bits. When INTCN is set to logic 1, then a match between the timekeeping registers and either of the alarm registers activates the $\overline{INT}$/SQW pin (if the alarm is enabled). Because the INTCN bit is set to logic 1 when power is first applied, the pin defaults to an interrupt output with alarms disabled. The pullup voltage can be up to 5.5V, regardless of the voltage on V_{CC}. If not used, this pin can be left unconnected.
4	$\overline{RST}$	Active-Low Reset. This pin is an open-drain input/output. It indicates the status of V_{CC} relative to the V_{PF} specification. As V_{CC} falls below V_{PF}, the $\overline{RST}$ pin is driven low. When V_{CC} exceeds V_{PF}, for t_{RST}, the $\overline{RST}$ pin is pulled high by the internal pullup resistor. The active-low, open-drain output is combined with a debounced pushbutton input function. This pin can be activated by a pushbutton reset request. It has an internal 50kΩ nominal value pullup resistor to V_{CC}. No external pullup resistors should be connected. If the oscillator is disabled, t_{REC} is bypassed and $\overline{RST}$ immediately goes high.
5–12	N.C.	No Connection. Must be connected to ground.
13	GND	Ground
14	V_{BAT}	Backup Power-Supply Input. When using the device with the V_{BAT} input as the primary power source, this pin should be decoupled using a 0.1µF to 1.0µF low-leakage capacitor. When using the device with the V_{BAT} input as the backup power source, the capacitor is not required. If V_{BAT} is not used, connect to ground. The device is UL recognized to ensure against reverse charging when used with a primary lithium battery. Go to www.maximintegrated.com/qa/info/ul.
15	SDA	Serial Data Input/Output. This pin is the data input/output for the I^2C serial interface. This open-drain pin requires an external pullup resistor. The pullup voltage can be up to 5.5V, regardless of the voltage on V_{CC}.
16	SCL	Serial Clock Input. This pin is the clock input for the I^2C serial interface and is used to synchronize data movement on the serial interface. Up to 5.5V can be used for this pin, regardless of the voltage on V_{CC}.

- **Timekeeping Registers**

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00h	0	10 Seconds			Seconds				Seconds	00–59
01h	0	10 Minutes			Minutes				Minutes	00–59
02h	0	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03h	0	0	0	0	0	Day			Day	1–7
04h	0	0	10 Date		Date				Date	01–31
05h	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06h	10 Year				Year				Year	00–99
07h	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08h	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09h	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0Ah	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0Bh	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0Ch	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 20 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0Dh	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0Eh	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0Fh	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11h	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12h	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

MPU6050 Datat Sheet

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0D	13	SELF_TEST_X	R/W	XA_TEST[4-2]			XG_TEST[4-0]				
0E	14	SELF_TEST_Y	R/W	YA_TEST[4-2]			YG_TEST[4-0]				
0F	15	SELF_TEST_Z	R/W	ZA_TEST[4-2]			ZG_TEST[4-0]				
10	16	SELF_TEST_A	R/W	RESERVED		XA_TEST[1-0]		YA_TEST[1-0]		ZA_TEST[1-0]	
19	25	SMPLRT_DIV	R/W	SMPLRT_DIV[7:0]							
1A	26	CONFIG	R/W	-	-	EXT_SYNC_SET[2:0]			DLPF_CFG[2:0]		
1B	27	GYRO_CONFIG	R/W	-	-	-	FS_SEL [1:0]		-	-	-
1C	28	ACCEL_CONFIG	R/W	XA_ST	YA_ST	ZA_ST	AFS_SEL[1:0]				
23	35	FIFO_EN	R/W	TEMP _FIFO_EN	XG _FIFO_EN	YG _FIFO_EN	ZG _FIFO_EN	ACCEL _FIFO_EN	SLV2 _FIFO_EN	SLV1 _FIFO_EN	SLV0 _FIFO_EN
24	36	I2C_MST_CTRL	R/W	MULT _MST_EN	WAIT _FOR_ES	SLV_3 _FIFO_EN	I2C_MST _P_NSR	I2C_MST_CLK[3:0]			
25	37	I2C_SLV0_ADDR	R/W	I2C_SLV0 _RW	I2C_SLV0_ADDR[6:0]						
26	38	I2C_SLV0_REG	R/W	I2C_SLV0_REG[7:0]							
27	39	I2C_SLV0_CTRL	R/W	I2C_SLV0 _EN	I2C_SLV0 _BYTE_SW	I2C_SLV0 _REG_DIS	I2C_SLV0 _GRP	I2C_SLV0_LEN[3:0]			
28	40	I2C_SLV1_ADDR	R/W	I2C_SLV1 _RW	I2C_SLV1_ADDR[6:0]						
29	41	I2C_SLV1_REG	R/W	I2C_SLV1_REG[7:0]							
2A	42	I2C_SLV1_CTRL	R/W	I2C_SLV1 _EN	I2C_SLV1 _BYTE_SW	I2C_SLV1 _REG_DIS	I2C_SLV1 _GRP	I2C_SLV1_LEN[3:0]			
2B	43	I2C_SLV2_ADDR	R/W	I2C_SLV2 _RW	I2C_SLV2_ADDR[6:0]						
2C	44	I2C_SLV2_REG	R/W	I2C_SLV2_REG[7:0]							
2D	45	I2C_SLV2_CTRL	R/W	I2C_SLV2 _EN	I2C_SLV2 _BYTE_SW	I2C_SLV2 _REG_DIS	I2C_SLV2 _GRP	I2C_SLV2_LEN[3:0]			
2E	46	I2C_SLV3_ADDR	R/W	I2C_SLV3 _RW	I2C_SLV3_ADDR[6:0]						
2F	47	I2C_SLV3_REG	R/W	I2C_SLV3_REG[7:0]							
30	48	I2C_SLV3_CTRL	R/W	I2C_SLV3 _EN	I2C_SLV3 _BYTE_SW	I2C_SLV3 _REG_DIS	I2C_SLV3 _GRP	I2C_SLV3_LEN[3:0]			
31	49	I2C_SLV4_ADDR	R/W	I2C_SLV4 _RW	I2C_SLV4_ADDR[6:0]						
32	50	I2C_SLV4_REG	R/W	I2C_SLV4_REG[7:0]							
33	51	I2C_SLV4_DO	R/W	I2C_SLV4_DO[7:0]							
34	52	I2C_SLV4_CTRL	R/W	I2C_SLV4 _EN	I2C_SLV4 _INT_EN	I2C_SLV4 _REG_DIS	I2C_MST_DLY[4:0]				
35	53	I2C_SLV4_DI	R	I2C_SLV4_DI[7:0]							
36	54	I2C_MST_STATUS	R	PASS_ THROUGH	I2C_SLV4 _DONE	I2C_LOST _ARB	I2C_SLV4 _NACK	I2C_SLV3 _NACK	I2C_SLV2 _NACK	I2C_SLV1 _NACK	I2C_SLV0 _NACK
37	55	INT_PIN_CFG	R/W	INT_LEVEL	INT_OPEN	LATCH _INT_EN	INT_RD _CLEAR	FSYNC_ INT_LEVEL	FSYNC _INT_EN	I2C _BYPASS _EN	-
38	56	INT_ENABLE	R/W	-	-	-	FIFO _OFLOW _EN	I2C_MST _INT_EN	-	-	DATA _RDY_EN
3A	58	INT_STATUS	R	-	-	-	FIFO _OFLOW _INT	I2C_MST _INT	-	-	DATA _RDY_INT

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
3B	59	ACCEL_XOUT_H	R	ACCEL_XOUT[15:8]							
3C	60	ACCEL_XOUT_L	R	ACCEL_XOUT[7:0]							
3D	61	ACCEL_YOUT_H	R	ACCEL_YOUT[15:8]							
3E	62	ACCEL_YOUT_L	R	ACCEL_YOUT[7:0]							
3F	63	ACCEL_ZOUT_H	R	ACCEL_ZOUT[15:8]							
40	64	ACCEL_ZOUT_L	R	ACCEL_ZOUT[7:0]							
41	65	TEMP_OUT_H	R	TEMP_OUT[15:8]							
42	66	TEMP_OUT_L	R	TEMP_OUT[7:0]							
43	67	GYRO_XOUT_H	R	GYRO_XOUT[15:8]							
44	68	GYRO_XOUT_L	R	GYRO_XOUT[7:0]							
45	69	GYRO_YOUT_H	R	GYRO_YOUT[15:8]							
46	70	GYRO_YOUT_L	R	GYRO_YOUT[7:0]							
47	71	GYRO_ZOUT_H	R	GYRO_ZOUT[15:8]							
48	72	GYRO_ZOUT_L	R	GYRO_ZOUT[7:0]							
49	73	EXT_SENS_DATA_00	R	EXT_SENS_DATA_00[7:0]							
4A	74	EXT_SENS_DATA_01	R	EXT_SENS_DATA_01[7:0]							
4B	75	EXT_SENS_DATA_02	R	EXT_SENS_DATA_02[7:0]							
4C	76	EXT_SENS_DATA_03	R	EXT_SENS_DATA_03[7:0]							
4D	77	EXT_SENS_DATA_04	R	EXT_SENS_DATA_04[7:0]							
4E	78	EXT_SENS_DATA_05	R	EXT_SENS_DATA_05[7:0]							
4F	79	EXT_SENS_DATA_06	R	EXT_SENS_DATA_06[7:0]							
50	80	EXT_SENS_DATA_07	R	EXT_SENS_DATA_07[7:0]							
51	81	EXT_SENS_DATA_08	R	EXT_SENS_DATA_08[7:0]							
52	82	EXT_SENS_DATA_09	R	EXT_SENS_DATA_09[7:0]							
53	83	EXT_SENS_DATA_10	R	EXT_SENS_DATA_10[7:0]							
54	84	EXT_SENS_DATA_11	R	EXT_SENS_DATA_11[7:0]							
55	85	EXT_SENS_DATA_12	R	EXT_SENS_DATA_12[7:0]							
56	86	EXT_SENS_DATA_13	R	EXT_SENS_DATA_13[7:0]							
57	87	EXT_SENS_DATA_14	R	EXT_SENS_DATA_14[7:0]							
58	88	EXT_SENS_DATA_15	R	EXT_SENS_DATA_15[7:0]							
59	89	EXT_SENS_DATA_16	R	EXT_SENS_DATA_16[7:0]							
5A	90	EXT_SENS_DATA_17	R	EXT_SENS_DATA_17[7:0]							
5B	91	EXT_SENS_DATA_18	R	EXT_SENS_DATA_18[7:0]							
5C	92	EXT_SENS_DATA_19	R	EXT_SENS_DATA_19[7:0]							
5D	93	EXT_SENS_DATA_20	R	EXT_SENS_DATA_20[7:0]							
5E	94	EXT_SENS_DATA_21	R	EXT_SENS_DATA_21[7:0]							
5F	95	EXT_SENS_DATA_22	R	EXT_SENS_DATA_22[7:0]							
60	96	EXT_SENS_DATA_23	R	EXT_SENS_DATA_23[7:0]							
63	99	I2C_SLV0_DO	R/W	I2C_SLV0_DO[7:0]							
64	100	I2C_SLV1_DO	R/W	I2C_SLV1_DO[7:0]							
65	101	I2C_SLV2_DO	R/W	I2C_SLV2_DO[7:0]							
66	102	I2C_SLV3_DO	R/W	I2C_SLV3_DO[7:0]							

Addr (Hex)	Addr (Dec.)	Register Name	Serial I/F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
67	103	I2C_MST_DELAY_CTRL	R/W	DELAY_ES_SHADOW	-	-	I2C_SLV4_DLY_EN	I2C_SLV3_DLY_EN	I2C_SLV2_DLY_EN	I2C_SLV1_DLY_EN	I2C_SLV0_DLY_EN
68	104	SIGNAL_PATH_RESET	R/W	-	-	-	-	-	GYRO_RESET	ACCEL_RESET	TEMP_RESET
6A	106	USER_CTRL	R/W	-	FIFO_EN	I2C_MST_EN	I2C_IF_DIS	-	FIFO_RESET	I2C_MST_RESET	SIG_COND_RESET
6B	107	PWR_MGMT_1	R/W	DEVICE_RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		
6C	108	PWR_MGMT_2	R/W	LP_WAKE_CTRL[1:0]		STBY_XA	STBY_YA	STBY_ZA	STBY_XG	STBY_YG	STBY_ZG
72	114	FIFO_COUNTH	R/W	FIFO_COUNT[15:8]							
73	115	FIFO_COUNTL	R/W	FIFO_COUNT[7:0]							
74	116	FIFO_R_W	R/W	FIFO_DATA[7:0]							
75	117	WHO_AM_I	R	-	WHO_AM_I[6:1]						-