



# **CANSAT INTEGRATION USING MICROCONTROLLER**

by

Muhammad Hafiz Bin Amir

Final project report submitted in partial fulfilment of  
the requirements for the  
Bachelor of Engineering (Hons)  
(Electrical and Electronics Engineering)

JUNE 2010

Universiti Teknologi PETRONAS  
Bandar Seri Iskandar  
31750 Tronoh  
Perak Darul Ridzuan

## **CERTIFICATION OF APPROVAL**

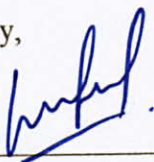
### **CANSAT INTEGRATION USING MICROCONTROLLER**

by

Muhammad Hafiz Bin Amir

A project dissertation submitted to the  
Electrical & Electronics Engineering Programme  
Universiti Teknologi PETRONAS  
in partial fulfilment of the requirement for the  
Bachelor of Engineering (Hons)  
(Electrical & Electronics Engineering)

Approved by,



---

(Dr. Mohamad Naufal Bin Mohamad Saad)

Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS  
TRONOH, PERAK

June 2010

## **CERTIFICATION OF ORIGINALITY**

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



---

Muhammad Hafiz Bin Amir



## ABSTRACT

Satellites are objects that are placed into orbit, among the types of satellites are astronomical, communication, earth observation and weather satellites. CanSat is a nano-scale satellite model with all the basic functions of a satellite, such as power subsystems, communication subsystems, attitude determination and control subsystem and payload subsystem which includes temperature and pressure monitoring sensors. This research emphasizes on the operation of CanSat which involves monitoring the temperature and pressure of the surrounding environment where the data can be transmitted almost instantaneously on demand. The satellites monitoring is done by using the accelerometer to determine the direction heading and the pressure sensor to determine the altitude. This data is processed by the microcontroller and the data will be encrypted to be passed through the MU-1 transmitter, then transmitted and received by the MU-1 receiver at the Ground Station. The design process involves developing the test circuit to understand the functionality of the sensors followed by integration of the satellite subsystems. The results acquired for several streaming measurements captured for a certain period of time is plotted to analyze the sensors performance and accuracy level.

## ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Mohamad Naufal Bin Mohamad Saad, for his continuous supports, guidance, encouragement and concerns throughout the whole process of making this thesis possible.

My appreciation also goes to Universiti Teknologi PETRONAS especially Electrical and Electronics Engineering Department lecturers and staff alike, for the aid and support given for me to excel in theoretical and technical works.

I would also like to express my greatest appreciation to Mr. Isnani Alias, Lab Technologist, Universiti Teknologi PETRONAS and all my course mate that play an important role in this project. Without their guidance and valuable information, this thesis would not be completed in time. Last but not least, thanks to my friends and family who have been supporting me throughout this Final Year Project.

# TABLE OF CONTENT

CERTIFICATION OF APPROVAL.....	ii
CERTIFICATION OF ORIGINALITY .....	iii
ABSTRACT.....	iv
ACKNOWLEDGEMENTS .....	v
LIST OF TABLES .....	viii
LIST OF FIGURES .....	ix
LIST OF ABBREVIATION .....	xi
CHAPTER 1 INTRODUCTION .....	1
1.1 Background of Study.....	1
1.2 Problem Statement .....	2
1.3 Objective.....	3
1.4 Scope Of Study .....	3
CHAPTER 2 LITERATURE REVIEW .....	5
2.1 Satellite Overview .....	5
2.2 Primary Components.....	8
2.2.1 Microcontroller (PIC16F877A).....	8
2.2.2 Temperature sensor (LM35DT) .....	10
2.2.3 Pressure sensor (MPX4115A) .....	11
2.2.4 Wireless Transmitter and Receiver (MU-1) .....	12
2.2.5 Accelerometer (ADXL330) .....	16
2.3 Subsystem Configuration .....	16

<b>CHAPTER 3 METHODOLOGY .....</b>	<b>19</b>
3.1 Procedure Identification .....	19
3.2 Tools and Equipment Required .....	21
<b>CHAPTER 4 RESULTS AND DISCUSSIONS .....</b>	<b>22</b>
4.1 Results .....	22
4.1.1 Feature identification of microcontroller.....	22
4.1.2 LM35 Temperature Sensor Development.....	23
4.1.3 MU-1 Wireless Transceiver Development .....	26
4.1.4 CanSat Ground Station Graphic Interface Software Development .....	37
4.2 Discussions .....	41
<b>CHAPTER 5 CONCLUSION AND RECOMMENDATIONS .....</b>	<b>43</b>
5.1 Conclusion .....	43
5.2 Recommendations .....	43
<b>REFERENCES.....</b>	<b>44</b>
<b>APPENDICES.....</b>	<b>46</b>
<b>APPENDIX A FIRST PROTOTYPE OF CANSAT.....</b>	<b>47</b>
<b>APPENDIX B FINAL PROTOTYPE OF CANSAT .....</b>	<b>48</b>
<b>APPENDIX C MICROCONTROLLER ALGORITITHM FOR                 CANSAT.....</b>	<b>49</b>
<b>APPENDIX D VISUAL BASIC 2008 ALGORITHM FOR MISSION                 CONTROL CENTER (MCC) .....</b>	<b>57</b>



## LIST OF TABLES

Table 1: Link parameters description.....	15
Table 2: Voltage reading based on test conditions .....	25
Table 3: MU-1 and microcontroller development process.....	26
Table 4: Signal Strength versus distance for MU-1 .....	36
Table 5: MU-1 and GS graphic interface software development process.....	37



## LIST OF FIGURES

Figure 1: CanSat simulation setup .....	7
Figure 2: Pin diagram for PIC16F877A microcontroller .....	8
Figure 3: LM35 temperature sensor interface circuit.....	11
Figure 4: Pin connection between MU-1 and PIC16F877 .....	14
Figure 5: Link parameter values .....	16
Figure 6: CanSat subsystem relationship .....	17
Figure 7: Flow Diagram for Project Work .....	19
Figure 8: Flow diagram for Timer0 application .....	22
Figure 9: Simulated oscilloscope reading for Timer0 application.....	23
Figure 10: The hardware and features of the microcontroller .....	24
Figure 11: Temperature test software flow.....	24
Figure 12: Data conversion from microcontroller to computer.....	27
Figure 13: HyperTerminal configuration parameters .....	27
Figure 14: Algorithm for 19200 bps data rate calibration.....	28
Figure 15: HyperTerminal view for 19200 bps data rate calibration.....	29
Figure 16: Algorithm for microcontroller control command transmission test.....	30
Figure 17: HyperTerminal view for microcontroller control command test.....	30
Figure 18: HyperTerminal view for MU-1 response command test.....	31
Figure 19: HyperTerminal view for MU-1 link setup response command test.....	32
Figure 20: Ground Station side data transmission test.....	33
Figure 21: Satellite side data reception test.....	33
Figure 22: Data conversion from microcontroller to computer through MU-1.....	34
Figure 23: Microcontroller algorithm flow diagram for two way communication .....	35

Figure 24: MU-1 two way communications at the Ground Station side through built in graphic interface software .....	35
Figure 25: Signal strength signature registered with the received data at the Ground Station .....	37
Figure 26: Overall design view of GS graphic interface software.....	38
Figure 27: GS graphic interface software flow diagram .....	39
Figure 26: CanSat functional diagram .....	41

## LIST OF ABBREVIATION

CanSat	Can Satellite
TT&C	Telemetry Tracking and Command
C&DH	Command and Data Handing
EPS	Electrical Power Subsystem
AC	Alternating Current
DC	Direct Current
RAM	Random Access Memory
EEPROM	Electronically Erasable Programmable Read Only Memory
ADC	Analogue-to-Digital
PWM	Pulse Width Modulation
LED	Light Emitting Diode
UART	Universal Asynchronous Receive/Transmit
USB	Universal Serial Bus
ASCII	American Standard Code for Information Interchange
RS-232	Recommended Standard 232
USART	Universal Serial Asynchronous Receive Transmit
MSB	Most Significant Bit
LSB	Least Significant Bit
NRZ	Non-Return-to-Zero



# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Background of Study**

Information is passed around ever more efficient nowadays especially in the form of digital data. In terms of hardware, one of the fastest ways to transmitting data is using fibre optic cables. Another method is by using wireless transmission. However for transmission, the digital data will have to be converted to analogue data and then combining with the carrier frequency for transmission. Stations at ground level with antennas are able to receive and transmit data among each other. To enhance this wireless ability, satellites play a role to further improve the speed and area coverage in wireless transmission. Satellites are objects that are placed into orbit. Among the types of satellites are astronomical, communication, earth observation and weather satellites. Astronomical satellites are used for observation of distant outer space objects while communication satellites are stationed in space for the purpose of telecommunications. Meanwhile weather satellites are used to monitor Earth's weather and climate and Earth observation satellites are intended for non-military uses such as environmental monitoring, meteorology and map-making [1].

With the availability of highly capable satellites that have high performance with cheaper cost and weight enables Malaysia to further strengthen the venture to do business in the space industry [2]. The use of small satellites will revolutionize the space industry as has the laptop progressively dominated the computer industry over the bulky personal desktop computer.

Among the endeavour of Malaysia in the space race to reach the orbit is through the National Space Agency (ANGKASA) with the introduction of the space education program that is SiswaSAT Program similar to the “CanSat” program in Japan and United States [3]. CanSat is a nano-scale satellite model with all the basic functions of a satellite, the subsystems of the satellite are fitted into a soda can of 325 millilitres. The CanSat will have nearly identical features to a real satellite in terms of the electronic circuitry, CanSat has its wireless communication system, sensors and central processing unit. In terms of the mechanical CanSat shall have its structure developed in terms of structural rigidity and the ability to be recovered after flight operation.

## **1.2 Problem Statement**

Space technology is a major strategic tool at present and in the future due to the unique global capabilities that it brings. It will help society overcome several threats to the quality of life on earth. In that sense space technology is fundamental to sustain security and one aspect of space technology is remote sensing [4].

Among the design considerations of satellites related to electronics is how the various subsystems operate such as the operation of the payload sensors which could be for example a temperature sensor, operation of the wireless transmitter and receiver at the ground station, integration of all the subsystems and the power consumption of the satellite.

Since environmental monitoring of the Earth at great heights is unfeasible with the application of aircrafts. Therefore, the solution is by using satellites, CanSat will be able to monitor the temperature and pressure of the surrounding environment at a specific height while the data can be transmitted almost instantaneously on demand. The satellites height is confirmed by monitoring the pressure reading from the pressure sensor and the rate of descent.



### **1.3 Objective**

The main objectives of this study are as follows.

1. To develop and manufacture the prototype CanSat that will perform the environmental monitoring with the use of temperature monitoring sensor and pressure monitoring sensor.
2. To develop the satellites positional system for confirmation of the location of the satellite by gathering data from the accelerometer and pressure sensor as part of the satellite's payload.

### **1.4 Scope Of Study**

The scope for this research involves identifying the features of the microcontroller such as timers, Analogue-To-Digital (ADC) converter and communication protocols such as the Universal Asynchronous Receive/Transmit (UART) that will be used to interface with the other primary components of the satellite. Simulations will be done to familiarize with the coding language.

The next procedure is the construction and individual testing of the circuit for the separate main electronic components such as the pressure sensor, temperature sensor, accelerometer and wireless transmitter and receiver. The individual tests are as follows:

- Temperature sensor
  - Constructing the biasing circuit and testing the output voltage to confirm with the relation between temperature and voltage value as stated in the datasheet.
  - Interfacing the output voltage of the temperature sensor with the ADC input of the microcontroller. The microcontroller will output the ADC values as duty cycles for the Pulse-Width-Modulation (PWM) that will be

connected to a Light Emitting Diode (LED). An algorithm for the microcontroller will have to be developed.

- Pressure sensor

- Constructing the biasing circuit and interfacing with the microcontroller. An algorithm for the microcontroller will have to be developed to verify the relationship between barometric pressure and voltage.

- Wireless transmitter and receiver

- Constructing the biasing circuit and interfacing with the microcontroller. The transmitter is tested by verifying that the receiver and the transmitted signal, the verification is done by connecting the output of the receiver to a serial converter circuit that enables serial communication with the computer.

- Accelerometer

- Constructing the biasing circuit and testing the output voltage to confirm with the relation between acceleration and voltage value.
- Interfacing the output voltage of the acceleration sensor with the ADC input of the microcontroller. The microcontroller will output the ADC values as duty cycles for the PWM that will be connected to a Light Emitting Diode (LED). An algorithm for the microcontroller will have to be developed.

## **CHAPTER 2**

### **LITERATURE REVIEW**

#### **2.1 Satellite Overview**

Among the subsystem for a satellite includes Control Subsystem, Telemetry Tracking and Command (TT&C), Command and Data Handling (C&DH) and Electrical Power System (EPS) [4]. Control Subsystem functions to stabilize the satellite and orient it in the desired direction during missions where there are external disturbance torques acting on the satellite. Control Subsystem also determines the satellites attitude using sensors and controls using actuators. Control Subsystem is coupled to propulsion and navigation subsystems. On the other hand, TT&C provides interface between spacecraft and ground systems. Payload mission data and spacecraft housekeeping data pass from spacecraft through this subsystem to operators and users at the operation centre. Operator commands also pass to the spacecraft through this subsystem to control the spacecraft and to operate the payload.

The C&DH functions to receive, validate, decode and distribute commands to other spacecraft systems and gathers, processes, and formats spacecraft housekeeping and mission data for downlink or use by an onboard computer. Finally, since all the subsystems described previously consist of electronic circuits that require power to operate. Therefore, the EPS provides, stores, distributes and controls a satellites electrical power. Among the power subsystem functions are:



- Supply continuous source of electrical power to spacecraft loads during mission life
- Control and distribute electrical power to the spacecraft
- Support power requirements for average and peak electrical load
- Provide converters for Alternating Current (AC) and regulated Direct Current (DC) power buses, if required
- Provide command and telemetry capability for EPS health and status, as well as control by ground station or an autonomous system
- Protect the payload against failures within the EPS

The environmental test for the CanSat involves two different tests. However, due to the lack of time provided for this research the experiments was unable to be conducted. The first test is to monitor the surrounding temperature and pressure, this will be done by placing the CanSat at a height of 10 meters from the ground level for an extended period of time from morning to evening, the setup for the test is described in Figure 1. CanSat will be operating throughout the entire operation where the readings from the temperature sensor, accelerometer and pressure sensor will be processed by the microcontroller and then transmitted to the Ground Station via the MU-1 transceiver's located on the CanSat and at the Ground Station. A real time graph will be plotted and all the data gathered will be recorded and saved for further analysis. All the readings will be observed through the computer via the Ground Station interface software. By conducting this test ensures the reliability of the CanSat operation for prolonged periods.

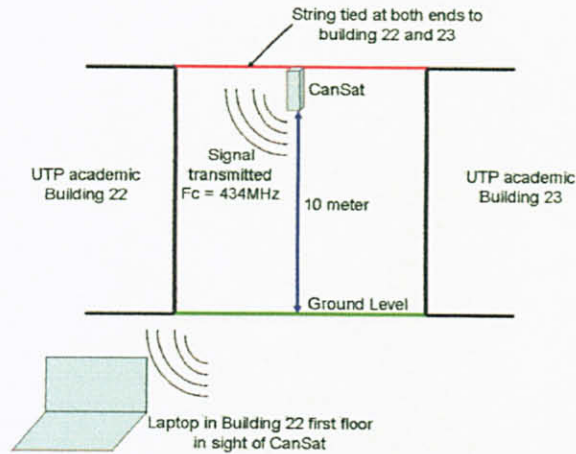


Figure 1: CanSat environmental test

The second test is to observe the operation of the operation of the accelerometer since for the first test the CanSat is stationary and the objective is for reliability. The second test will observe the readings of the X, Y and Z-axis readings of the accelerometer. The setup for the test is shown in Figure 2. The operation of the CanSat will be in the identical manner as the previous environmental test only this time the expected result should be a spike in the value for the accelerometer while the temperature sensor and the pressure sensor is constant. All the readings will be observed through the computer via the Ground Station interface software.

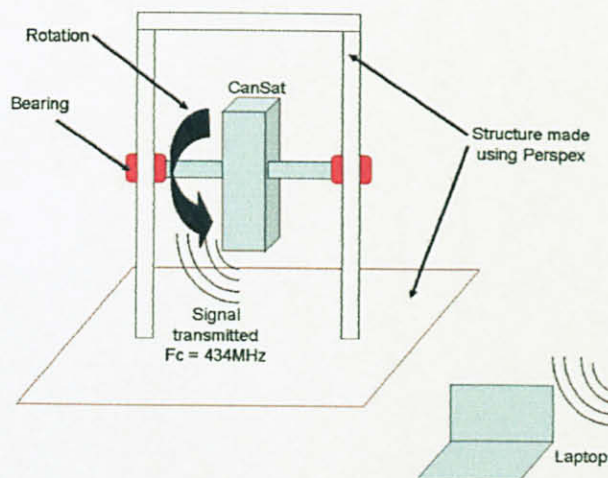


Figure 2: CanSat accelerometer operation test



## 2.2 Primary Components

To design the prototype CanSat with similar subsystems settings as a satellite would have, the primary components involved in developing the prototype CanSat are as follows:

### 2.2.1 Microcontroller (PIC16F877A)

The Microchip PIC16F877A microcontroller is one of the most popular microcontrollers in the market, the pin diagram is shown in Figure 2.

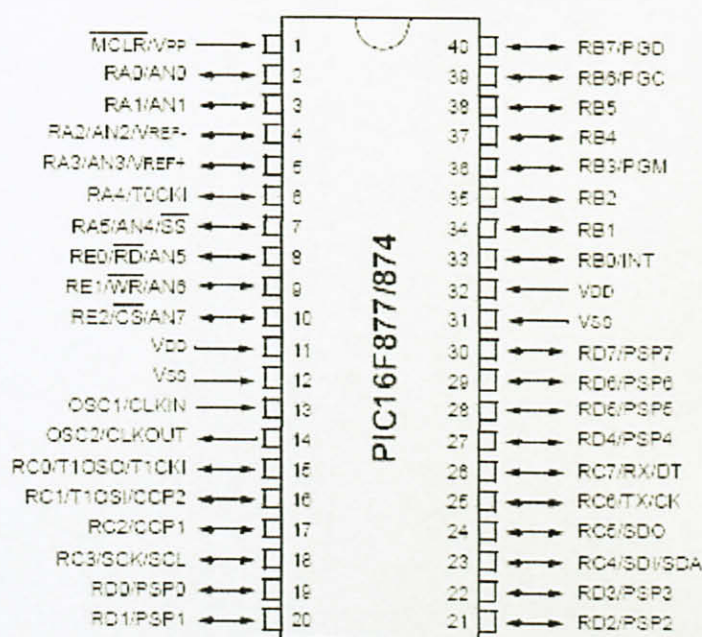


Figure 3: Pin diagram for PIC16F877A microcontroller [11]

The PIC16F877A Microcontroller includes eight kilo bytes of internal flash Program Memory, together with a large Random Access Memory (RAM) area and an internal Electronically Erasable Programmable Read Only Memory (EEPROM). An 8-

channel 10-bit ADC is also included within the microcontroller, making it ideal for real-time systems and monitoring applications [11]. The first step is to understand the individual features of the PIC16F877A microcontroller that will be used to integrate the sensors and communications subsystems of CanSat. Therefore the programs are designed using MPLAB software and compiled using PIC C Compiler and the generated HEX file is uploaded into PIC Simulator IDE. The features that are to be simulated are as follows.

- Timers (Timer0, Timer1 and Timer2)
- ADC
- Serial communication

Timer0 is usually an 8-bit timer/counter created around an 8-bit counter unit. Hence its count range is restricted from 0 to 255. It is most often used as a time or tick.

Meanwhile, Timer1 is a 16-bit timer that can be used in exactly the same manner as Timer0. However, Timer1 is actually designed to be used in an entirely different fashion. Where Timer0 usually accomplishes its tasks by reloading or resetting the timer as needed to produce the delays, Timer1 is designed to keep running and never be reset or reloaded. Timer1 may be used alone or with the capture/compare/PWM module. The PWM function works the same with either Timer1 or Timer2.

Timer2 is involved with the capture and compare modes of operation. Capture mode is used to capture the contents of Timer1 when an external event occurs on PortC bit 2 (RC2). Meanwhile, the Compare mode is a related function in that it uses the contents of the Timer1 registers to determine when to cause an output event on RC2. Pulse-width modulation (PWM) is a variation of compare mode [11].

The Analogue to Digital converter in the microcontroller is a 10-bit converter that can be used to convert analogue voltages at one of eight inputs. The analogue to digital conversion of the analogue input signal results in a corresponding 10-bit digital number. The A/D module has high and low voltage reference input. The results of the digitally



converted analogue input voltage levels are stored in a memory location ADRESH:ADRESL registers [11].

Universal Synchronous Asynchronous Receiver Transmitter (USART) is also another serial communication feature for the PIC16F877A and can be configured as full duplex asynchronous communication. UART uses the non-return-to-zero (NRZ) format with one START bit, eight or nine data bits and one STOP bit. An eight bit baud rate generator is used to generate the oscillator [11]. The UART transmits and receives the Least Significant Bit (LSB) first.

### 2.2.2 *Temperature sensor (LM35DT)*

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin, the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling.

The LM35 does not require any external calibration or trimming to provide typical accuracies of  $\pm 1/4^{\circ}\text{C}$  at room temperature and  $\pm 3/4^{\circ}\text{C}$  over a full  $-55$  to  $+150^{\circ}\text{C}$  temperature range [5] and the sensitivity of the temperature sensor will be set to  $10\text{mV}/^{\circ}\text{C}$ . The circuit for the LM35 is quite straightforward with only a parallel  $100\text{k}\Omega$  resistor at the voltage output pin to limit the current and a 5V supply at the voltage supply pin to bias the LM35. The temperature sensor will be part of the payload subsystem. The interface circuit for the LM35 is shown in Figure 4.

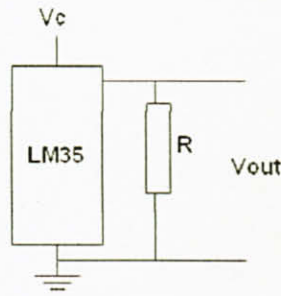


Figure 4: LM35 temperature sensor interface circuit

### 2.2.3 Pressure sensor (MPX4115A)

The MPX4115A series (115 kPa) absolute pressure sensor is suitable for the application of barometric pressure measurement and altimeter with the following advantages [6]:

- 15 kPa to 115 kPa measuring range
- Single +4.85V to 5.35V supply
- Ideally suited for microcontroller based systems
- Temperature compensated from -40°C to +125°C
- Durable Epoxy Unibody Element or Thermoplastic (PPS) Surface Mount Package

The pressure sensor will be part of the payload subsystem and also part of the ADCS subsystem due to the usefulness of the data since pressure values can also be used in determining altitude. The altitude calculation by using pressure value is related by the following equation:

$$\text{Altitude (m)} = -2616 \times \ln (\text{pressure}/101.325) \dots\dots\dots ( 1 )$$

#### 2.2.4 *Wireless Transmitter and Receiver (MU-1)*

The selected wireless transmitter and receiver is the MU-1 since it is a low power radio modem capable to be embedded with various interface circuits for transmission of serial data. Using a simple system of commands, the user can concentrate on designing the transmitting and receiving protocols for the data using the commands, without needing to be aware of the control system of the radio component. By using the Universal Asynchronous Receive/Transmit (UART) protocol interface with a single-chip CPU, or a RS232 format interface for computers as the interface for transmitting and receiving data and for issuing commands, it is possible for the user to develop systems quickly. In addition, it is designed to minimize design difficulties involving the high frequency components in the user system. Using the relay function it is also possible to gather data from and control equipment in remote locations that require several intermediate receiving stations before reaching the target station [7]. The wireless transmitter and receiver will be part of the communication subsystem.

The communication subsystem can be divided into several sub-components. It consists of the transmitter, communication medium and receiver. The characteristics of the communication subsystem are:

- Perform data transmission from the satellite to the ground station from at most 200m away
- Maintain data integrity with band-pass filtering and digital modulation techniques
- To communicate the status and position of the satellite to the ground station
- Utilizing 434 MHz armature radio frequency band for communication
- Communication shall convey at least 1kbps in full-duplex mode
- The baud rate is 19200bps



In this communication subsystem, the selected operating frequency is chosen to be 434 MHz; according to International Telecommunications Union (ITU) Band designations, 434 MHz falls under band number 9 with the designation of UHF (Ultra High Frequency). The length of the antenna corresponds to the wavelength of the transmitted signal; with some analysis the length of the antenna has been determined to be 0.18m (18cm). We will be using telescopic antenna for the system as it provides a more directive signal with higher transmitting power [9].

For the connection of the MU-1 at the Ground Station is basically via Universal Serial Bus (USB) connection as the module has a built in interface circuit. Meanwhile, for the interface with the microcontroller requires a proper interface circuit. The interface circuit to integrate MU-1 with the PIC16F877 microcontroller is shown in Figure 5. The descriptions for the pins of the MU-1 that will be used are used are as follows.

- RXD - Serial data receive terminal from the microcontroller
- TXD - Serial data transmit terminal to the microcontroller
- CTS - Hardware flow control signal to determine if the MU-1 can output data through the TXD pin on active low and is controlled by the microcontroller
- RTS - Hardware flow control signal to determine if the MU-1 can input data through the RXD pin on active low and is controlled by the microcontroller

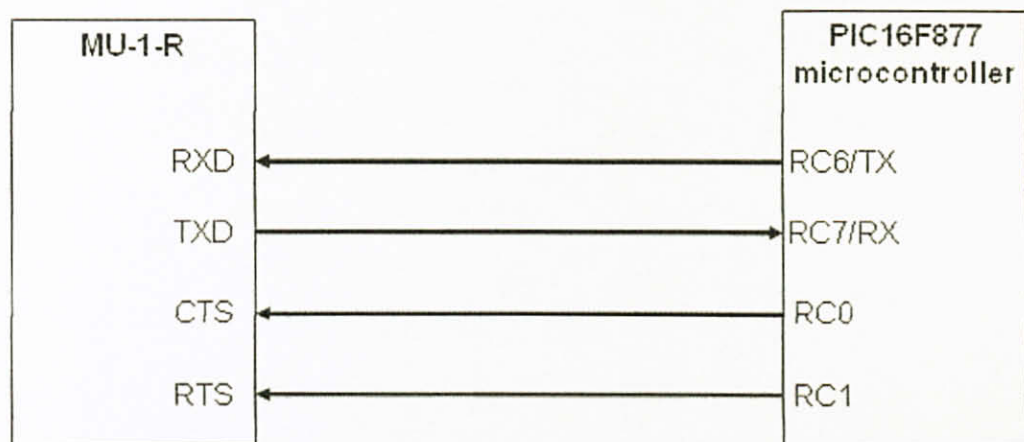


Figure 5: Pin connection between MU-1 and PIC16F877

To ensure successful command and response signal transmission between the MU-1 and the microcontroller the baud rate is set at 19200kbps. The type of command that the microcontroller can send to the MU-1 are Control Transmit Command which is used for system control and Data Transmit Command which is used for transmitting system data between the MU-1's.

Link parameters between the transmitting and receiving MU-1 need to be initialized with the specific Control Transmit Command. Table 1 describes the link parameters

Table 1: Link parameters description

Parameter	Description
Frequency channel	There are 64 values that can be set within the 434MHz band range, both MU-1 stations must have the same frequency channel
User ID	ID given to the MU-1 user for identification of the user system. Communication is not possible unless all equipment within the user system is set with the same User ID
Group ID	Set the same Group ID for all equipment within the group as identifications numbers when building other systems
Equipment ID	The Equipment ID differentiates the MU-1 of the transmitting and the receiving station, therefore the Equipment ID must be unique.
Destination ID	At the transmitting and receiving MU-1, the Equipment ID of the receiving MU-1 is entered.
Route Information	Indicates the route to the target which has the list of the Equipment IDs of the relay stations and Equipment ID of the target station.

There are various connections configuration for the MU-1 network when there are more than two MU-1's in use. However for the CanSat prototype there will only be two MU-1 units in use that is the transmitting MU-1 as part of the CanSat TT&C Subsystem and the receiving MU-1 as part of the Ground Station setup. This configuration is also called a 1 hop communication and to configure this setup both MU-1's must have the relay function disabled and this is done through the Route Information link parameter setup. Figure 6 shows the link parameter setup values that will be used.



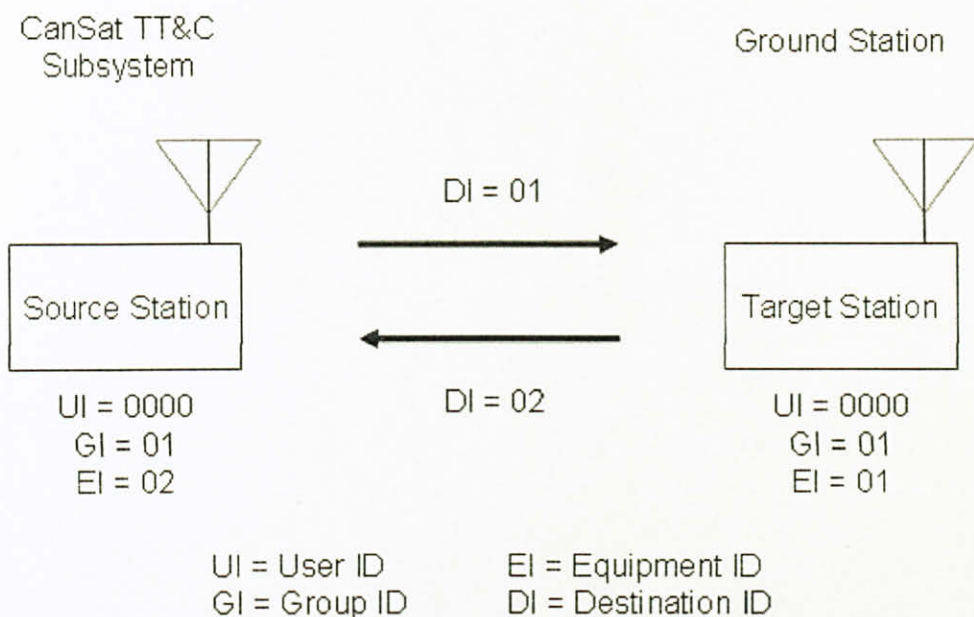


Figure 6: Link parameter values

### 2.2.5 Accelerometer (ADXL330)

The ADXL330 is a small, thin, low power, complete three axis accelerometer with signal conditioned voltage outputs, all on a single monolithic integrated circuit. Acceleration is measured with a minimum full-scale range of  $\pm 3$  g. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration [8]. The accelerometer will be part of the ADCS subsystem in determining the rate of movement as to provide external feedback to the microcontroller.

## 2.3 Subsystem Configuration

From the setup of a typical satellite configuration consisting of the Control, TT&C, C&DH and EPS subsystem. CanSat hardware configuration is related to each other as described in Figure 7.



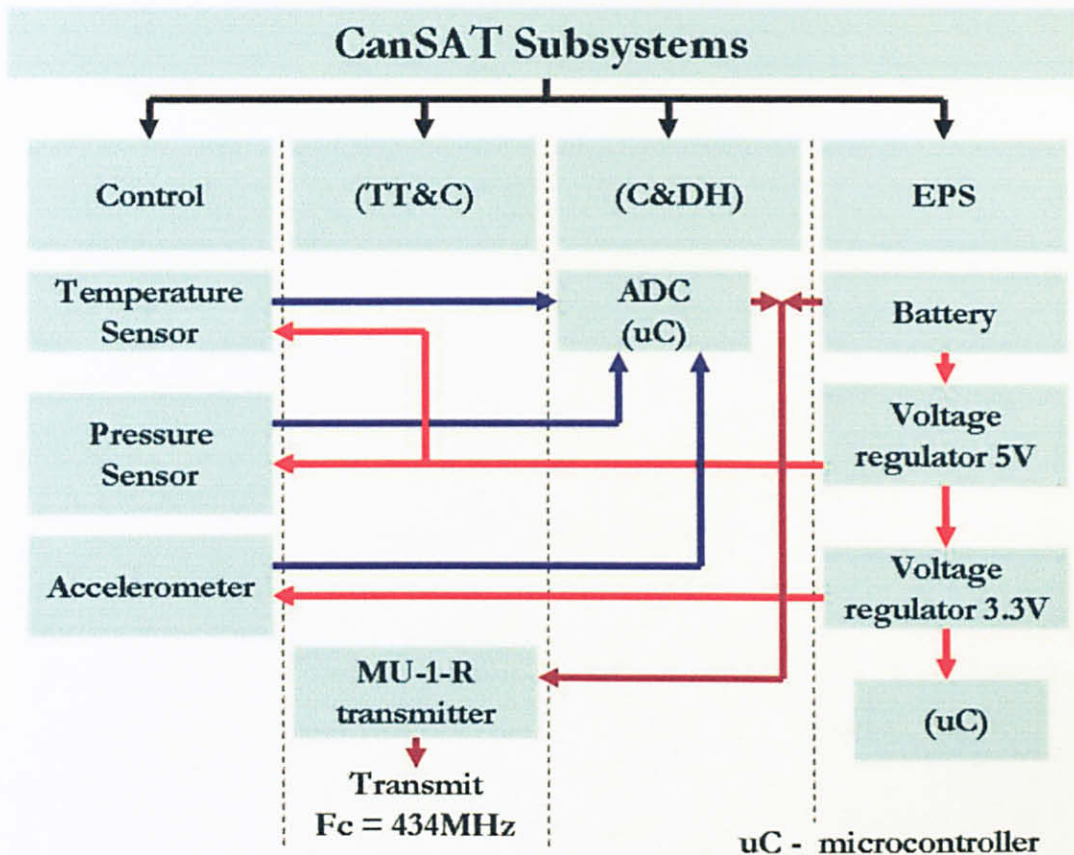


Figure 7: CanSat subsystem relationship

The EPS subsystem consists of the power supply for the satellite which will be powered by batteries rated at 9V. Since the temperature sensor requires a supply voltage of 5V therefore the battery terminals are connected to the 5V voltage regulator circuit and the output is then used to bias the temperature sensor and also to be stepped down by the 3V voltage regulator for the accelerometer, MU-1 transmitter, barometric pressure sensor and microcontroller supply voltage.

The Control subsystem shall include the sensors that are the temperature sensor, accelerometer and pressure sensor as the data obtained provides information of the satellites altitude, movement direction and temperature of the surrounding environment. Control subsystem is then linked to the C&DH subsystem when the data output of the sensors are processed by the microcontroller. The microcontroller will receive the

analogue voltage level output from the temperature sensor and accelerometer through the ADC port and convert into eight bits of digital data. Since the ADC ports cannot be triggered all at once, the sequence will be the ADC port for temperature followed by X-axis acceleration, Y-axis acceleration, Z-axis acceleration and finally the barometric pressure. An added feature of the accelerometer is the Self Test pin that is used to test if the accelerometer module is functioning correctly [8], this can be verified using the microcontroller.

Finally is the TT&C subsystem where the microcontroller will rearrange the eight bits of data which is the temperature, X-axis acceleration, Y-axis acceleration, Z-axis acceleration, and pressure sensor reading and will send the data to the MU-1 to for a sequence of transmissions until all the readings are transmitted and the loop transmits the next reading for the temperature and so on. On the other hand, in between transmissions the microcontroller will also poll for any incoming signals from the Ground Station to be passed to C&DH for further processing.

## CHAPTER 3

### METHODOLOGY

#### 3.1 Procedure Identification

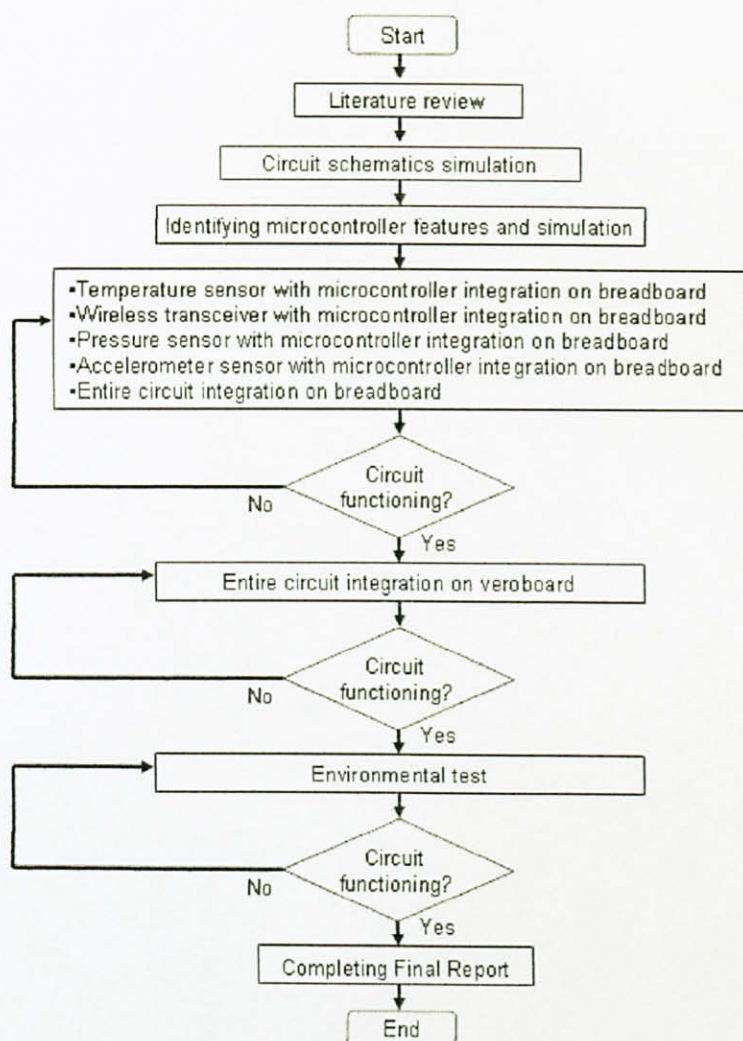


Figure 8: Flow Diagram for Project Work



Reference is done by referring to the previous technical documentations done from the participation of Universiti Teknologi PETRONAS (UTP) in the previous SiswaSAT competition and also referring to the technical documents from other universities and higher education centres which have also participated in the competition and relevant knowledge in satellite development. Hands-on technical aid is also gained by becoming a mentee to the participants from the UTP satellite team (AeroUTP) and also lecturers and technicians to gain knowledge on building electronic circuits and programming.

Among the necessary information that is required are the sensors that will be used and the number of sensors that will be implemented, other than that is to understand how the wireless communications between the satellite and Ground Station is established. Besides that, crucial information is essential for the programming to ensure that all the satellite subsystems and Ground Station can communicate effectively.

The approach done is by conducting the research and development of the prototype satellite at the same time.

### **3.2 Tools and Equipment Required**

The tools and equipment to be used for the studies are as follows.

1. MPLAB IDE 8.10
2. PIC C Compiler
3. PIC Simulator IDE
4. Microsoft Visual Basic 8.0
5. Accelerometer (ADXL330)
6. Pressure Sensor (MPX4115A)
7. Temperature Sensor (LM35DT)
8. PIC16F877A Microcontroller
9. Wireless Transmitter and Receiver (MU-1)
10. PICC Lite C Compiler

For the part one of the project research will be concentrating on identifying the features of the microcontroller, therefore extensive and familiarization of the software MPLAB IDE 8.10, PIC C Compiler and PIC simulator IDE is used. The temperature monitoring circuit is also developed in the first part involving the LM35DT. The second part of the research involves the remaining sensors and wireless transmission which is the Pressure Sensor (MPX4115A), Accelerometer (ADXL330), and Wireless Transmitter and Receiver (MU-1). The Ground Station development mainly involves the graphic interface software that will be developed using Microsoft Visual Basic 8.0.

## CHAPTER 4

### RESULTS AND DISCUSSION

#### 4.1 Results

Results are obtained from the analysis of two different tests which is to identify the features of the microcontroller. The second test is the individual testing of the temperature sensor.

##### 4.1.1 Feature identification of microcontroller

All the timers function similarly with each other in the microcontroller, therefore only Timer0 is simulated. A demonstration of this feature is as shown in Figure 9.

Ko

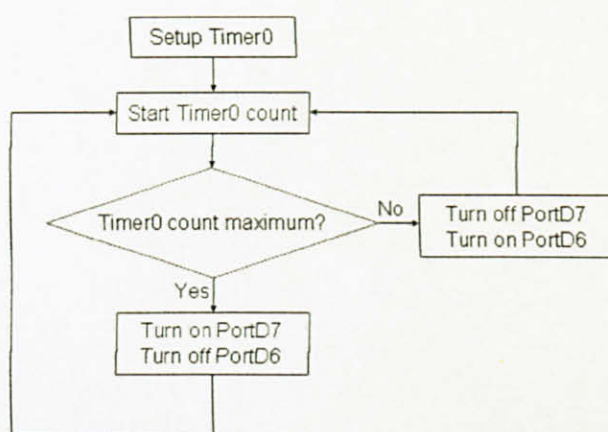


Figure 9: Flow diagram for Timer0 application



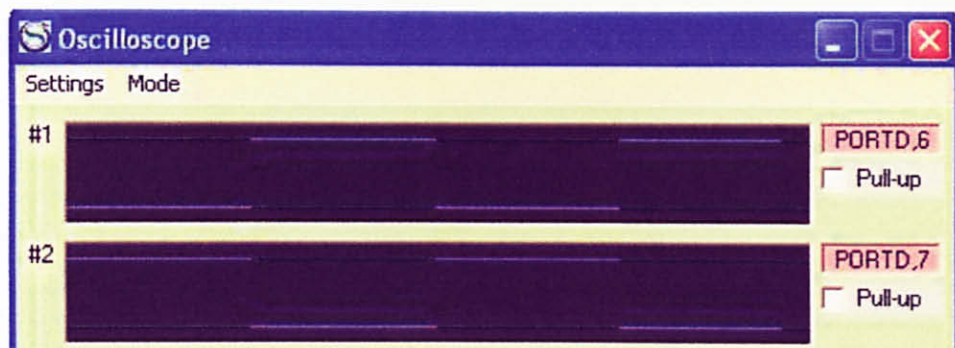


Figure 10: Simulated oscilloscope reading for Timer0 application

From Figure 10, it is observed that the states of the logic level at PortD6 and PortD7 changes as the count for Timer0 reaches maximum, the exact frequency of the changes cannot be observed for the software.

#### 4.1.2 LM35 Temperature Sensor Development

One of the essential components in the CanSat is the temperature sensor as part of the payload systems. Therefore, tests should be done to confirm that the readings of the temperature sensor correspond to the exact temperature. The flow diagram in terms of the hardware configuration and the microcontroller features that is used for the experiment and also the software flow diagram are shown in Figure 11 and Figure 12 respectively.

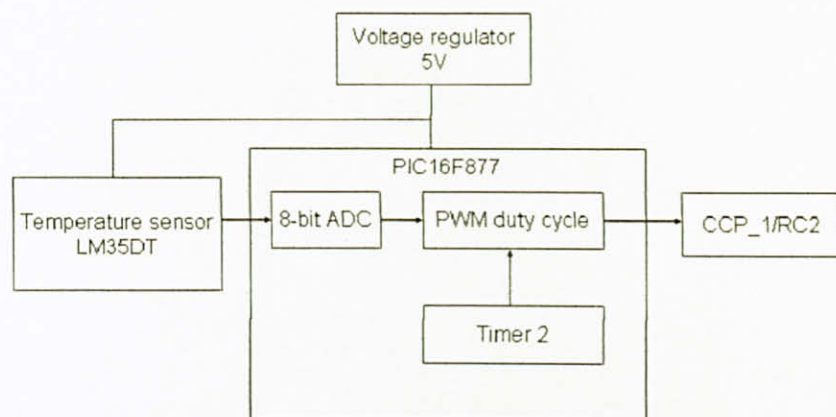


Figure 11: The hardware and features of the microcontroller

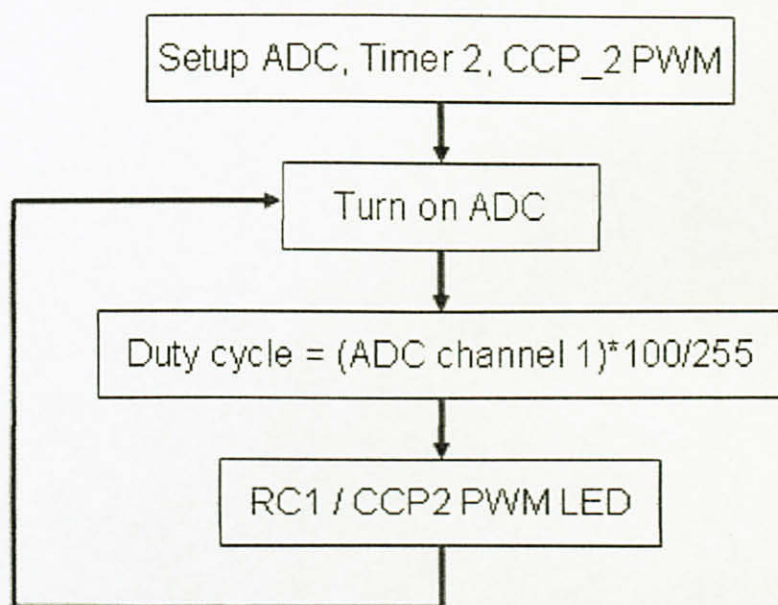


Figure 12: Temperature test software flow

The tests are conducted for various temperature levels to confirm that the temperature sensor (LM35DT) is operational with the following test conditions. The observed voltage readings are shown Table 2.

Table 2: Voltage reading based on test conditions

Test condition	Voltage reading (V)
Room temperature	0.284
Below room temperature	0.213
Above room temperature	0.370
Touching with finger	0.334

The method to test the temperature monitoring circuit at room temperature is straightforward and just turning on the circuit in standard conditions on the laboratory table and observing the reading of the voltage output level at the LM35DT and also observing the brightness of the LED. The voltage reading at the LM35DT voltage output terminal for this condition is 0.284V.

Meanwhile to test the sensor below room temperature is by turning on the circuit in a chilled environment and this test is conducted in the refrigerator and the reading in 0.213V, it is also observed that the LED brightness is dimmer than when the circuit is tested at room temperature. The next test is to obtain the voltage reading to be above the room temperature, the method chosen is to heat a metal object and then touch the metal with the body of the LM35DT and the voltage reading is observed to be 0.370V, the LED brightness for this test seems to be brighter than when the test is conducted at room temperature. Finally, an extra observation made is that by touching the body part of the LM35DT with our fingers, the voltage reading changes to 0.334V.

Since the other remaining sensors which is the accelerometer and the pressure sensor operate on the same principle with the temperature sensor that they all have analogue output voltage values that will be fed to the ADC input of the microcontroller. Therefore, the development of the accelerometer and pressure sensor is similar and the next procedure for the sensors is for the environmental test.



### 4.1.3 MU-1 Wireless Transceiver Development

UART protocol is used for communication between the MU-1 and the microcontroller, the development process for the integration with the satellite and Ground Station design is described in Table 3.

Table 3: MU-1 and microcontroller development process

Development process	Description
Microcontroller frequency calibration	Configuring HyperTerminal in the computer to be able to receive serial data at 19200 bps
Microcontroller UART transmission	Transmission of sequence of ASCII characters from microcontroller and viewed from HyperTerminal
MU-1 transmitter response test	Microcontroller transmission of control commands to MU-1 and response observed through HyperTerminal
MU-1 link parameter setup	Setting up the MU-1 with the microcontroller as receiver or transmitter
MU-1 data transmission	One way communication between Ground Station transceiver and satellite transceiver
Microcontroller data reception from MU-1	Processing of MU-1 data reception to the microcontroller and observing with HyperTerminal
Satellite and Ground Station communication	Two way communication between satellite and Ground Station

The first step in developing the integration system between the microcontroller and the MU-1 as described in Table 3 is to calibrate the data rate to 19200 bps which is the default data rate of the MU-1. To observe the ASCII characters as the microcontroller sweeps the data rate frequency range, the signals are observed through HyperTerminal from the computer. HyperTerminal is a built in Windows XP software which is an intermediate software to monitor peripheral data inputs from the various ports of the computer. To interface the microcontroller with the computer, an interface circuit is

constructed that converts UART signals to RS-232 protocol which is the communication format for the computer. Figure 13 describes the flow of data from the microcontroller to the computer and Figure 14 describes the settings for the HyperTerminal software to be able to process incoming data rate of 19200 bps and each frame shall consists of eight bits of data.

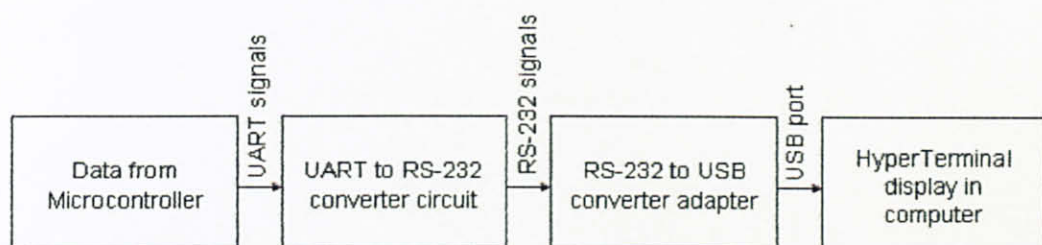


Figure 13: Data conversion from microcontroller to computer

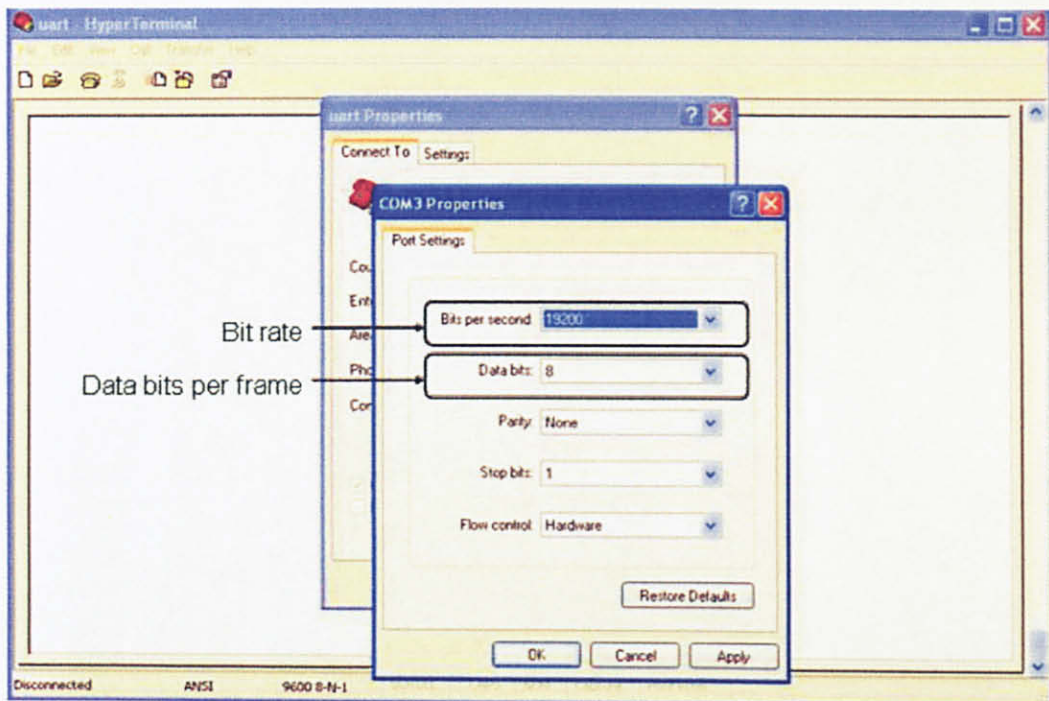


Figure 14: HyperTerminal configuration parameters

An algorithm is developed for the microcontroller to be able to transmit the predetermined ASCII sequence of characters which is "TEST". The flow diagram for the algorithm is described in Figure 15.

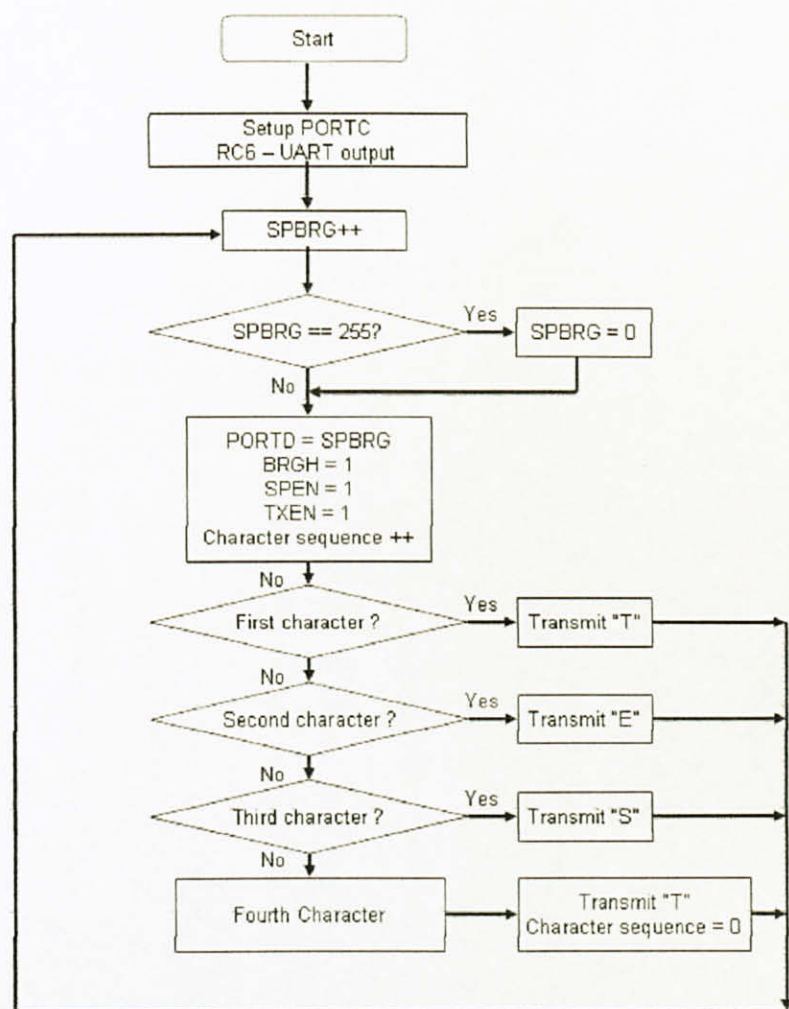


Figure 15: Algorithm for 19200 bps data rate calibration

From Figure 15, SPBRG is the data register in the microcontroller that is used to assign the data rate and since the values of SPBRG are output at PORTD where PORTD is connected to LED's, the value assigned to SPBRG can be determined as an adequate delay time is assigned for the period where each sequence of characters are transmitted. SPEN is to enable the peripherals outputs for the UART signals through PORTC, BRGH





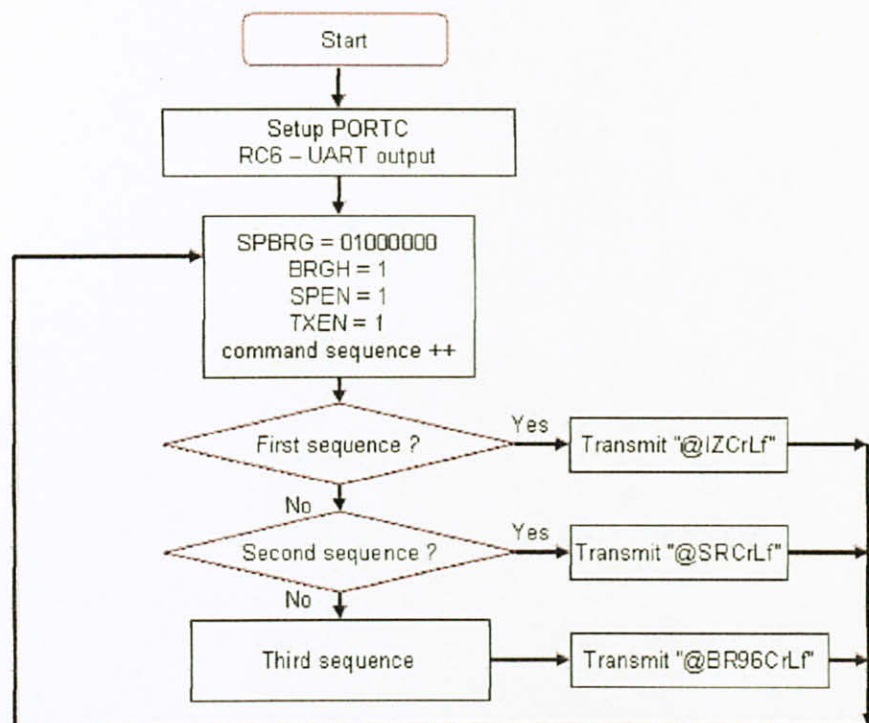


Figure 17: Algorithm for microcontroller control command transmission test

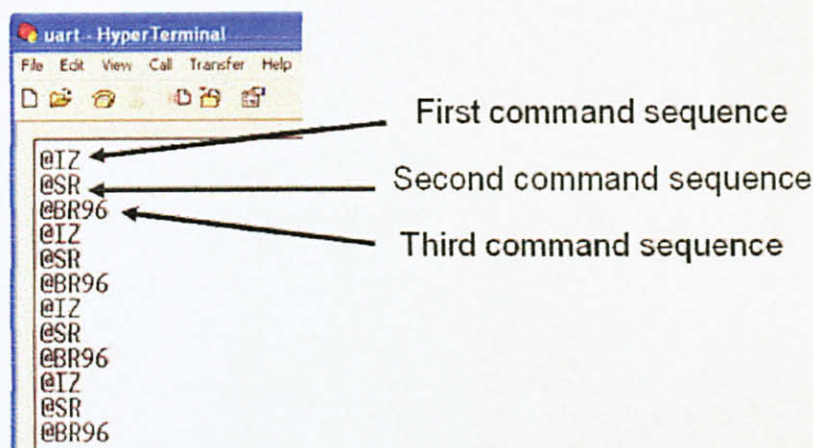


Figure 18: HyperTerminal view for microcontroller control command test

Following the control command transmission test from the microcontroller is the test to observe the response from the MU-1 therefore the UART to RS-232 circuit is receiving input signal from the MU-1 TX terminal instead of connected to the TX pin of

the microcontroller as in Figure 13. The microcontroller software algorithm for this test is the same as Figure 17 with a slight modification to include an adequate amount of delay time to allow the MU-1 to transmit the response command signals after each control command sequence from the microcontroller. The response from the MU-1 is observed in HyperTerminal and is shown in Figure 19.

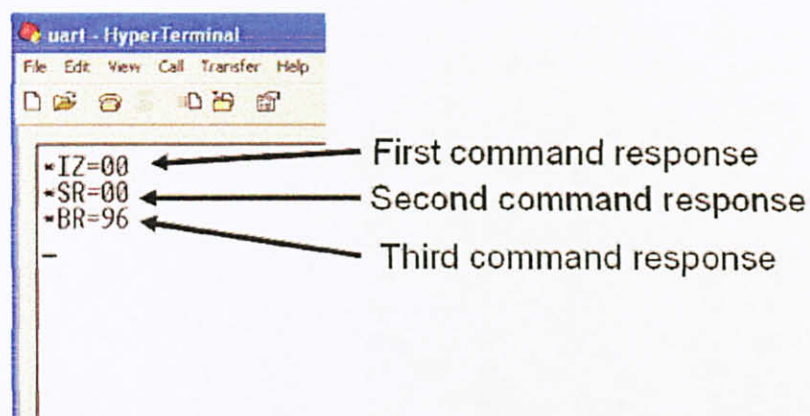


Figure 19: HyperTerminal view for MU-1 response command test

Since the previous tests where MU-1 retransmits control responses confirms that the microcontroller can transmit control commands to MU-1, the next step in the MU-1 development is to setup the link parameter for the MU-1 at the satellite side either as the receiver of the transmitter as described in Table 1 and the values will follow Figure 5 in Literature Review. The software algorithm flow resembles the flow in Figure 16 with changes made to the transmitted command sequence where the first command is "UI0000CrLf" followed by "GI01CrLf" and "EI01CrLf", the Destination ID is set with the Route Information parameter whereby if value is the same as the Equipment ID (EI) than that MU-1 unit becomes the receiver, otherwise the MU-1 becomes the transmitter. Therefore both values are tested to setup the MU-1 as the receiver followed by transmitter using the commands "RT01CrLf" and "RT02CrLf". The command response from the MU-1 is observed through HyperTerminal as shown in Figure 20.



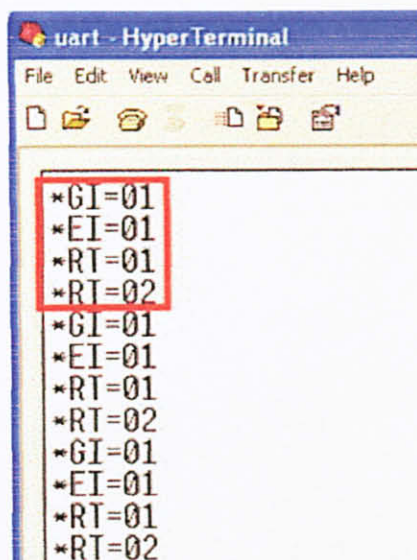


Figure 20: HyperTerminal view for MU-1 link setup response command test

The next procedure to confirm successful communication link is establish between both MU-1's at the satellite side and at the Ground Station side is to transmit data command signals from the Ground Station to the Satellite. The parameter link setup of the Ground Station MU-1 is done using the built in graphic interface software where the commands are issued in the Data Input Line [10] and the control command values follows the values in Figure 5 in Literature Review. Figure 21 is the view for the built in graphic interface software for the Ground Station meanwhile Figure 22 is the HyperTerminal view to observe the processes of the MU-1 at the satellite side.

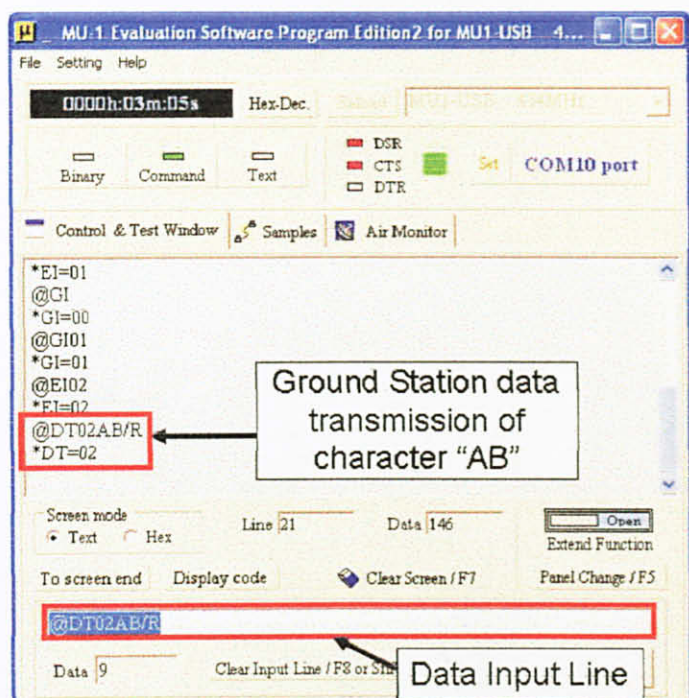


Figure 21: Ground Station side data transmission test

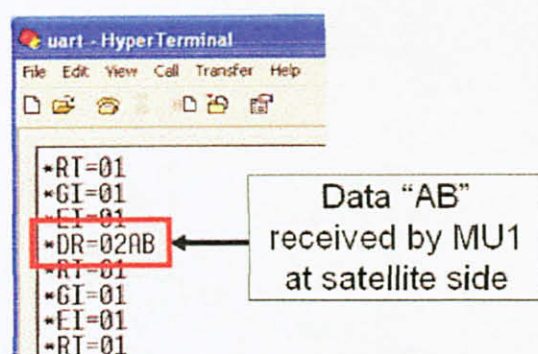


Figure 22: Satellite side data reception test

Since the satellite is considered to be in orbit, therefore the satellite requires a certain level of autonomous manoeuvring capabilities. Therefore the next procedure is to integrate the MU-1 and the microcontroller at the satellite side where the MU-1 is able to receive signals from the MU-1 for further processing. The hardware flow diagram from the microcontroller to the HyperTerminal software in the computer is shown in Figure 23.

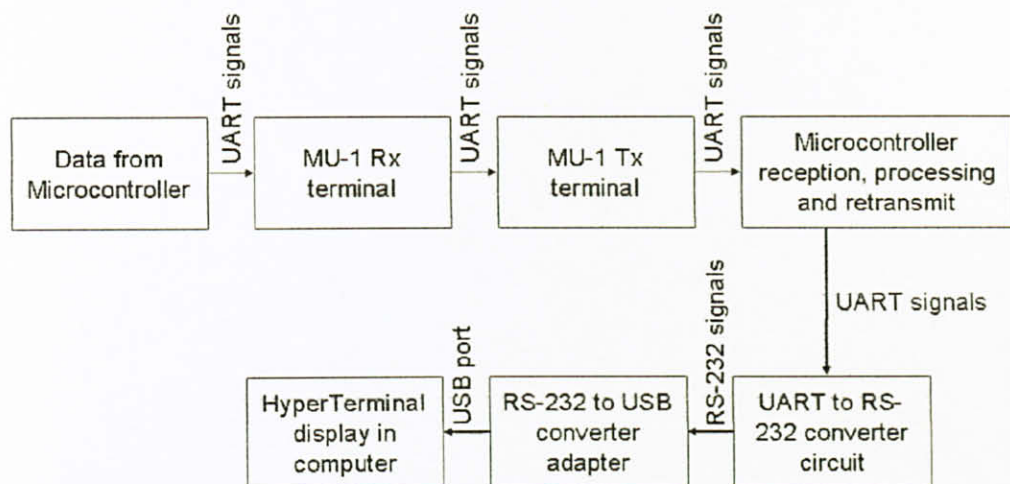


Figure 23: Data conversion from microcontroller to computer through MU-1

Finally in the MU-1 development process is to simulate the autonomous capabilities when the MU-1 at the satellite side is integrated with the microcontroller. The simulation begins as the satellite sends a data sequence “@DT05HELLO/RCrLf” to the Ground Station and repeats this process, if Ground Station replies “@DT03YES/RCrLf” and the microcontroller correctly oversamples the data sequence the response data to the Ground Station will be “@DT04GOOD/RCrLf”. Figure 24 shows the algorithm flow diagram for the microcontroller and Figure 25 shows the signal processes from the MU-1 at the Ground Station side via the built in graphic interface software.



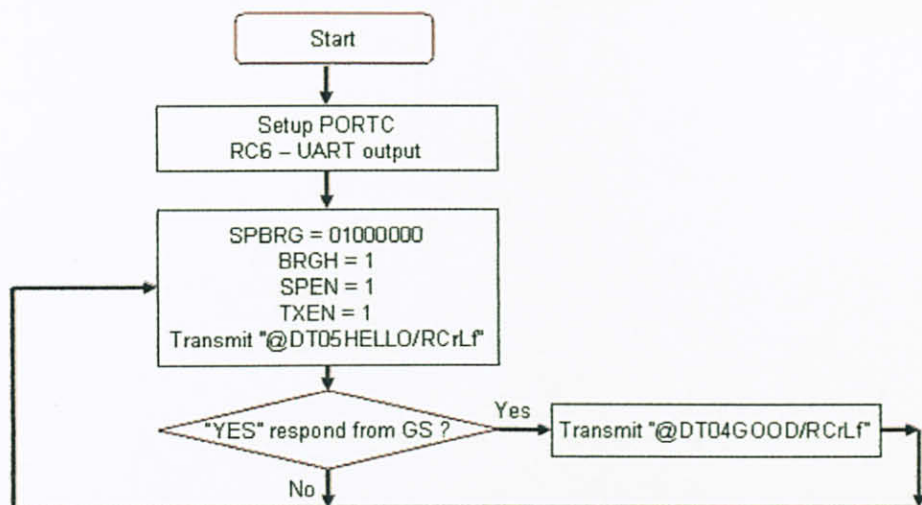


Figure 24: Microcontroller algorithm flow diagram for two way communication

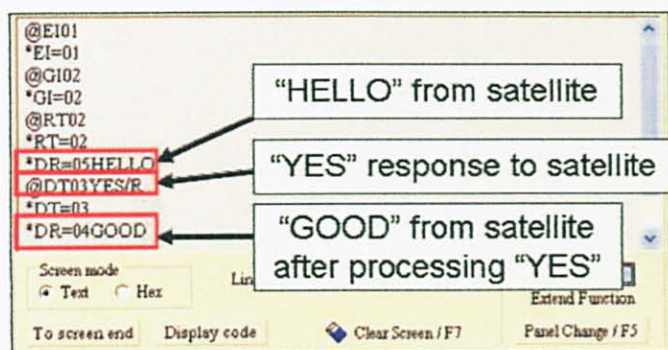


Figure 25: MU-1 two way communications at the Ground Station side through built in graphic interface software

Once all the basic command and transmit command sequence to operate the MU-1 have been understood, the capabilities of the MU-1 are tested in order to find the maximum range that the MU-1 can communicate between each other. A range test was conducted at a horizontal ground level to determine the maximum distance that the MU-1 can communicate with each other, for each increment there is an indication of the received signal strength attached with the received data and the distance was confirmed by visual inspection of the measured distance using a measuring tape. The software flow is the same as in Figure 23 with the modification being that every transmit signal is changed from ".../RCrLf" to "...%ACrLf", the character "%A" sends the command to

attach the signal strength at the beginning of the transmitted data before the actual data [7].

The received signal strength displayed at the built in graphic interface software for the MU-1 is shown in Table 4 and for the next increment at 117m no signal was received anymore which indicates that the transmitted signal from CanSat was too weak. An example of how the readings were taken is shown in Figure 26.

Table 4: Signal Strength versus distance for MU-1

Distance (meter)	Signal Strength (dB)
0	0x3A = 58
10.5	0x3F = 63
21	0x42 = 66
31.5	0x3B = 59
52.5	0x43 = 67
63	0x4C = 76
73.5	0x4D = 77
49	0x50 = 80
84	0x52 = 82
94.5	0x54 = 84
105	0x55 = 85

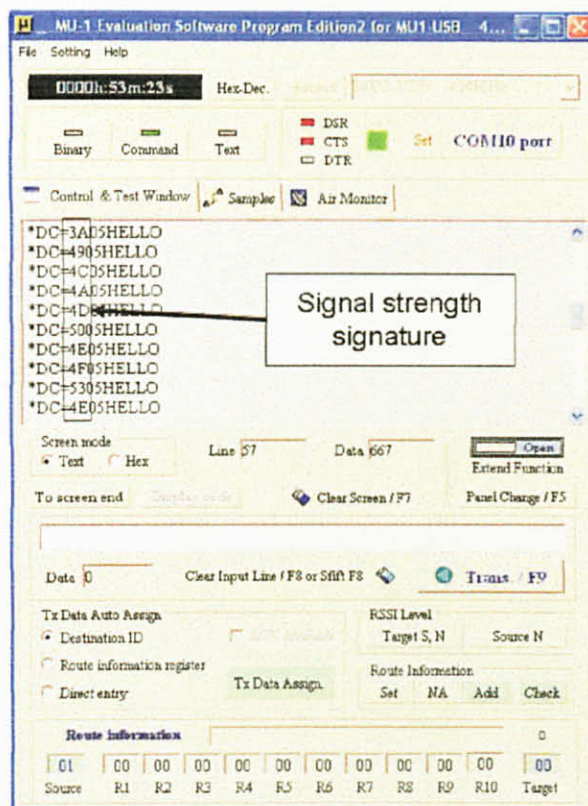


Figure 26: Signal strength signature registered with the received data at the Ground Station

#### 4.1.4 CanSat Ground Station Graphic Interface Software Development

UART protocol is used for communication between the MU-1 at the Ground Station (GS) and the computer where ASCII characters will be sent and received among the devices, the development process for the software is shown in Table 5.

Table 5: MU-1 and GS graphic interface software development process

Development process	Description
Configuring the GS MU-1 using the software	Setting the Group ID and observing the reply data signal from the MU-1
Receiving data from CanSat MU-1	Receiving and displaying the data from the temperature and accelerometer of the CanSat



Classifying the data from CanSat payload	Classifying the data for the temperature and 3-axis from the accelerometer and displaying in separate columns for each data representation
Plotting the graph for the classified data representation	Plotting the graph for each data representation on the same axis by converting the ASCII characters to decimal numbers

The overall design view of graphic interface for the GS software is shown in Figure 27 and the software flow diagram is shown in Figure 28.

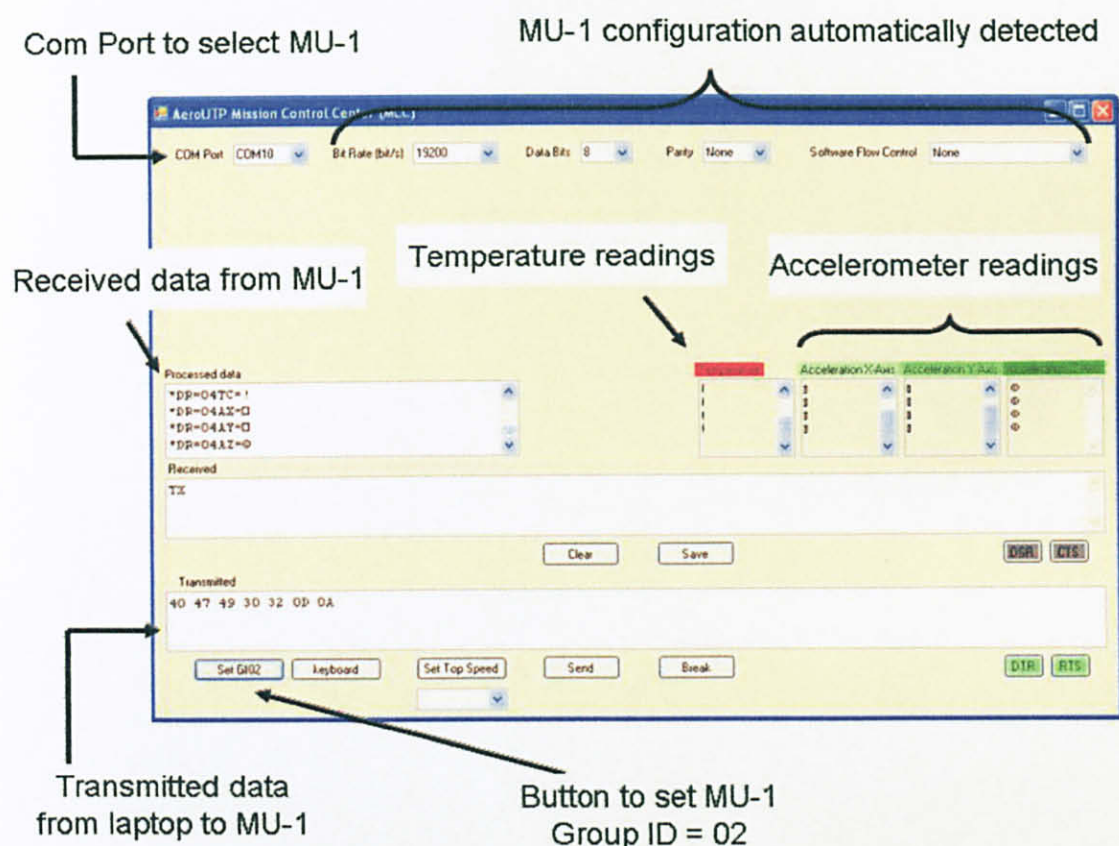


Figure 27: Overall design view of GS graphic interface software

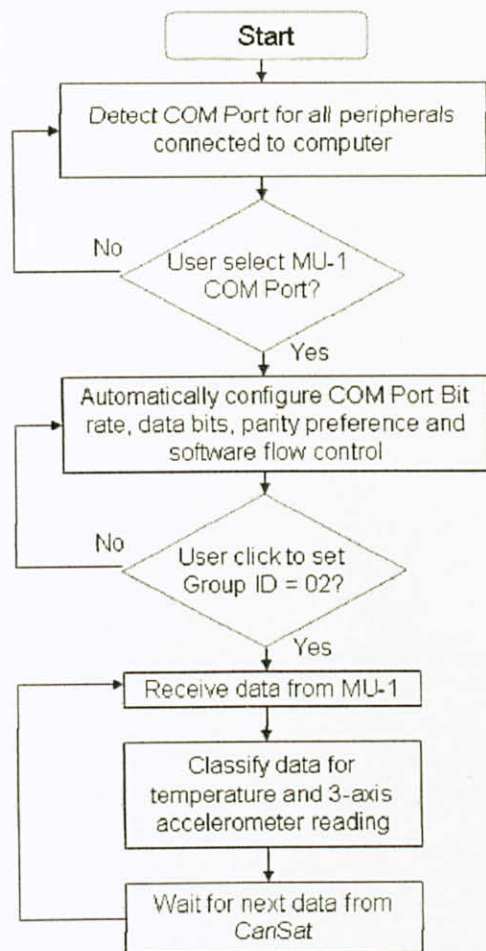


Figure 28: GS graphic interface software flow diagram

Upon executing the program AeroUTP MCC.exe from Figure 27, all the textboxes will be blank, the textboxes are the COM Port, Bit Rate (bit/s), Data Bits, Parity, and Software Flow Control. Once the selection of the appropriate port designated for the MU-1 is done, the other textboxes will be automatically configured with the software auto detection feature. Up to this process, all the other textboxes will still be empty and the next button to be clicked is the 'Set GI=02' and the textbox 'Processed Data' will display '\*GI=02' and this is the reply from the MU-1, Group ID is set to 02 since it is earlier defined for the MU-1 at the CanSat and this must be the same for successful communication [7]. The 'Transmitted' textbox will display '40 47 49 30 32 0D 0A' which is '\*GI02CrLf' in hexadecimal.

After a moment while waiting for the data from the CanSat sensors to be ready, the data will be received and the initial reception will be displayed in the 'Processed Data' textbox. The program processes each byte of data received to find the sequence code to classify the readings for the temperature and the three values for the three axis acceleration.

The software will scan all the sequence of ASCII characters received for the sequence 'TC' for temperature, 'AX' for the X-axis acceleration, 'AY' for the Y-Axis acceleration, 'AZ' for the Z-Axis acceleration. Upon detecting this sequence, the software will automatically assume the following ASCII characters after the character '=' are the readings. The readings will then be displayed in their respecting textboxes where the change in values can be observed.



4.2 Discussion

From the Results, it is observed that the features of the microcontroller that was used includes Timer0, Timer1, Timer2, Analogue-to-Digital converter and Serial Communication which is the UART protocol. The circuit diagram of the microcontroller that describes the connection between the sensors is shown in Figure 29.

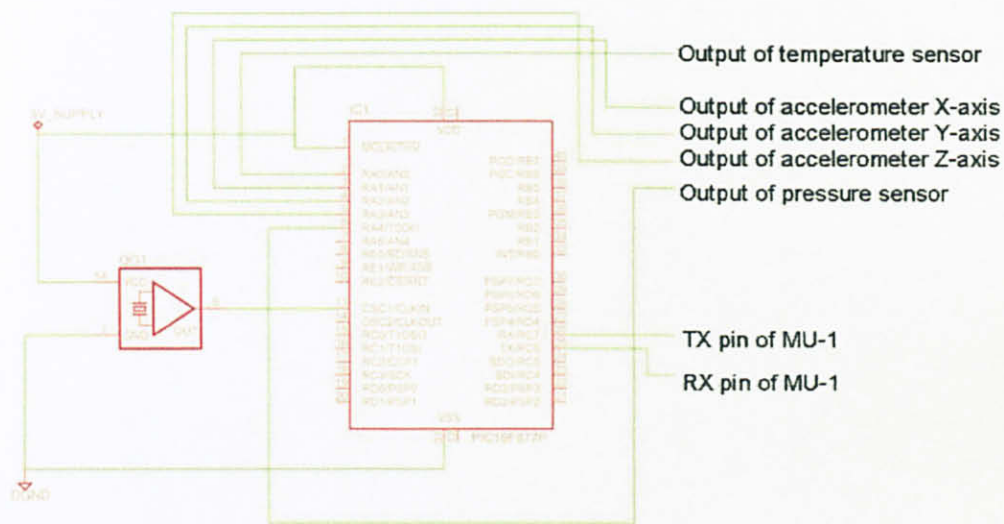


Figure 29: CanSat microcontroller schematic

The circuit in Figure 29 is used for the final environmental test of the CanSat. Two voltage regulators are used for the circuit which are 5V and 3V regulators, the circuit for each voltage regulator is shown in Figure 30 and the overall voltage regulator circuit for the CanSat is shown in Figure 31.

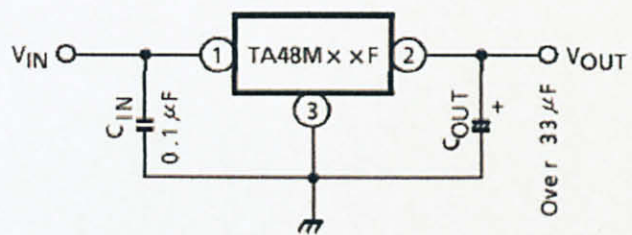


Figure 30: Voltage regulator schematic [12]

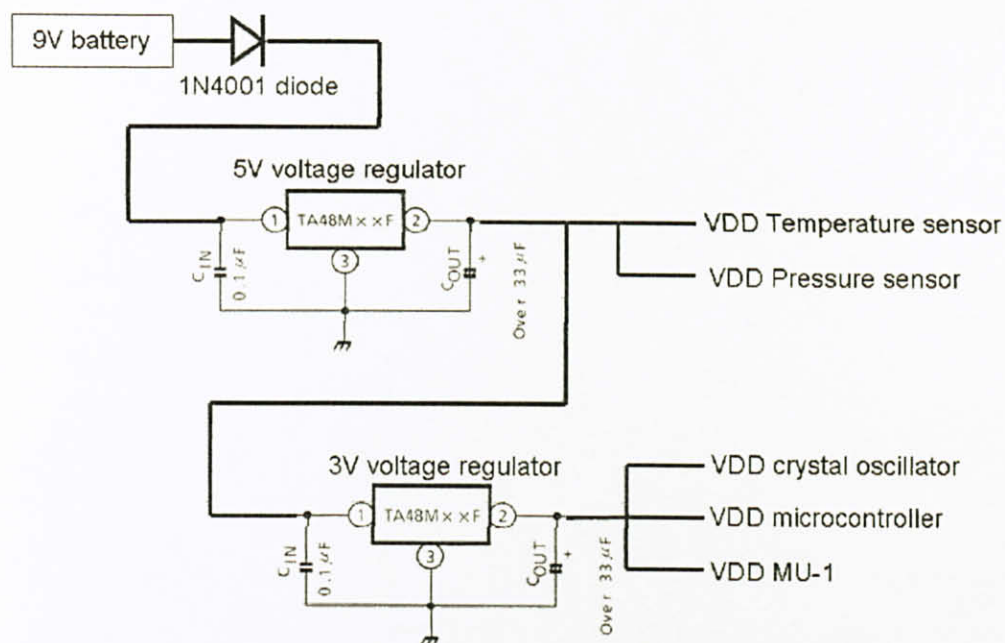


Figure 31: CanSat voltage regulator schematic

The Design of the CanSat requires the use of two voltage regulators is that the temperature sensor and pressure sensor has a typical operating voltage of 5V meanwhile the wireless transceiver MU-1 can only operate up to a typical voltage of 3V and since there is serial communication between the MU-1 and the microcontroller therefore the microcontroller and the crystal oscillator for the microcontroller need to be operating at the same voltage as the MU-1 which is at 3V. A concern is also raised if the output voltage for the temperature sensor and pressure sensor exceeds 3V, this is not a problem since it is calculated that both the sensors shall not have readings of more than 3V which is acceptable for the analogue input voltage for the microcontroller ADC to give an accurate reading. The value of the capacitors used for the voltage regulators is determined experimentally and found that the values used is suitable in eliminating any oscillating frequency and ensure constant DC voltage supply. A 1N4001 diode is also connected between the 9V battery and the input of the 5V voltage regulator to avoid any unwanted reverse bias voltages that could damage the circuit.

## **CHAPTER 5**

### **CONCLUSION AND RECOMMENDATION**

#### **5.1 Conclusion**

As a conclusion, the features of the PIC16F877 microcontroller are relevant to the development for the satellite development with the use of Timer0, Timer1, Timer2 and the Analog-to-Digital Converter. Whereby, Timer0, Timer1, Timer2 will be used for the timing sequence to arrange the storage of data flow starting from temperature, pressure and the three axis acceleration values finally followed by the transmission. The serial communication between the transceiver will be done using the UART feature. The test conducted on the temperature sensor also proved the functionality of LM35DT in correctly matching the voltage value to the exact temperature reading.

#### **5.2 Recommendation**

Future development plans for the CanSat integration involves redeveloping the circuit in the Printed Circuit Board format. Extra features to further develop the CanSat into a more functional satellite include the integration between a digital camera and global positioning satellite receiver. Finally, the mechanical structure can be developed for the CanSat and to perform the simulation at the specific height with the use of helium filled balloon tethered by a string to the ground to ensure the balloon does not cause the satellite to drift away.



## REFERENCE

- [1] Committee on the Peaceful uses of Outer Space (2008), Report of the United Nations/Russian Federation/European Space Agency Workshop on the Use of Microsatellite Technologies for Monitoring the Environment and Its Impact on Human Health, Programs and projects on space science and technology, pp. 6.
- [2] Azreena Ahmad (2006), Malaysia CanSat Program, The Program, pp 5 – 11.
- [3] Ibrahim Seeni Mohd, Mazlan Hashim, Samsudin Ahmad (2004), Research Strategies for remote sensing development in Malaysia, Current Status of Remote Sensing Applications, pp. 2 – 3.
- [4] Ahmad Sabirin Arshad (2004), Small Satellites Technology Development in Malaysia, TiungSAT-1 Program, pp. 13-20.
- [5] National Semiconductor Corporation (2000), LM35 Precision Centigrade Temperature Sensors, Features, pp. 1.
- [6] Motorola Freescale Semiconductor Inc (2001), MPX4115A Series, Integrated Silicon Pressure Sensor for Manifold Absolute Pressure, Altimeter or Barometer Applications On-Chip Signal Conditioned, Temperature Compensated and Calibrated, Features, pp. 1.
- [7] Circuit Design Inc (2006), Embedded Low Power Radio Modem MU-1-R 434MHz Operation Guide, How to Use the MU-1, pp 14 – 24.

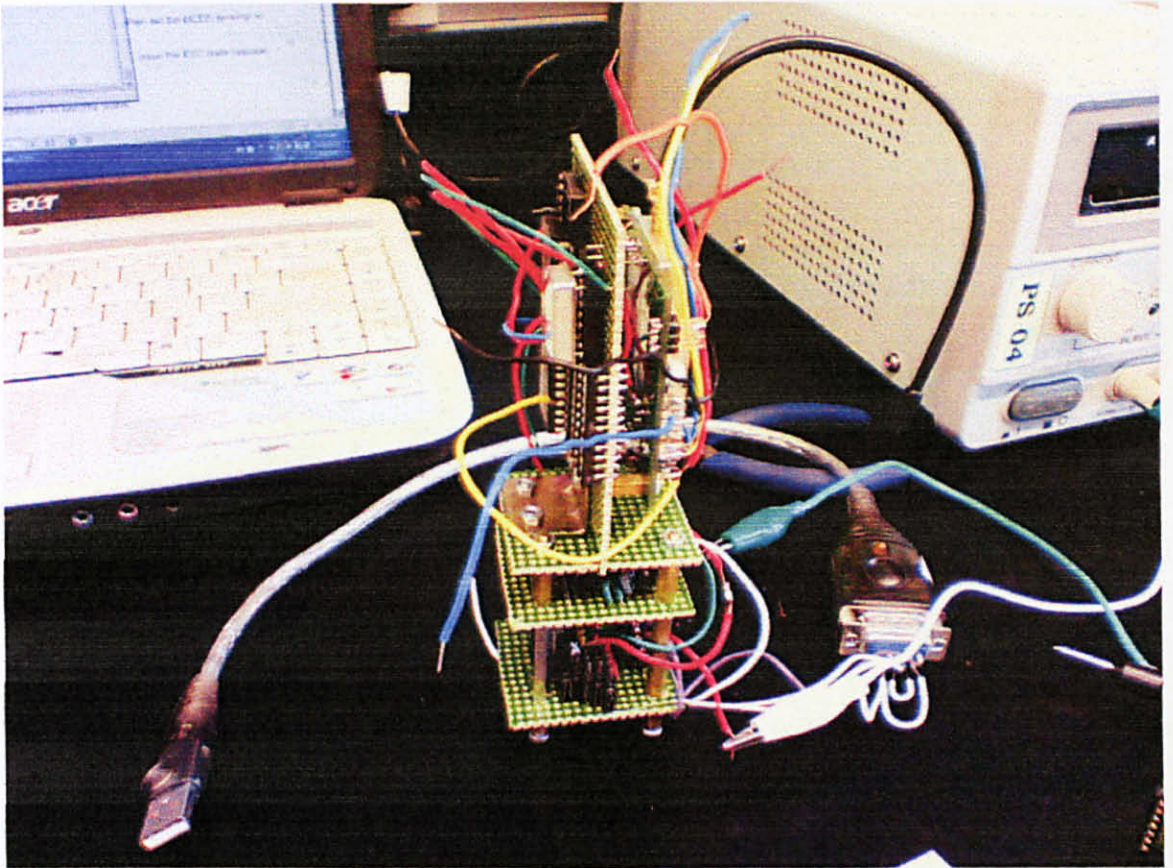
- [8] Analog Devices Inc. (2006), Small, Low Power, 3-Axis,  $\pm 3g$  iMEMS® Accelerometer, Feature, pp. 1.
  
- [9] Tan Chew Huoy, Er Mee Hoon, Faridah bt. Zainul Hashimi, Muhd. Zaiful Asmi Sulaiman, Mohamad Fakhruruddin Romeli (2008), Malaysia SiswaSat Competition 2008 UKM Team Critical Design Review, SiswaSat Design and Development, pp. 12 – 17.
  
- [10] Circuit Design Inc. (2005), Embedded Low Power Radio Modem MU-1 Evaluation Software Program Operation Guide, Explanation of the Control and Test Windows, pp. 7 – 15.
  
- [11] Microchip Technology Inc. (2001), PIC16F87X Data Sheet 28/40-Pin 8-Bit CMOS FLASH Microcontrollers, Device Overview, pp. 5 – 10.
  
- [12] Toshiba (1999), Bipolar Liner Integrated Circuit Silicon Monolithic TA48M0XXF Three Terminal Low Dropout Voltage Regulator, Features, pp. 1.

## APPENDICES



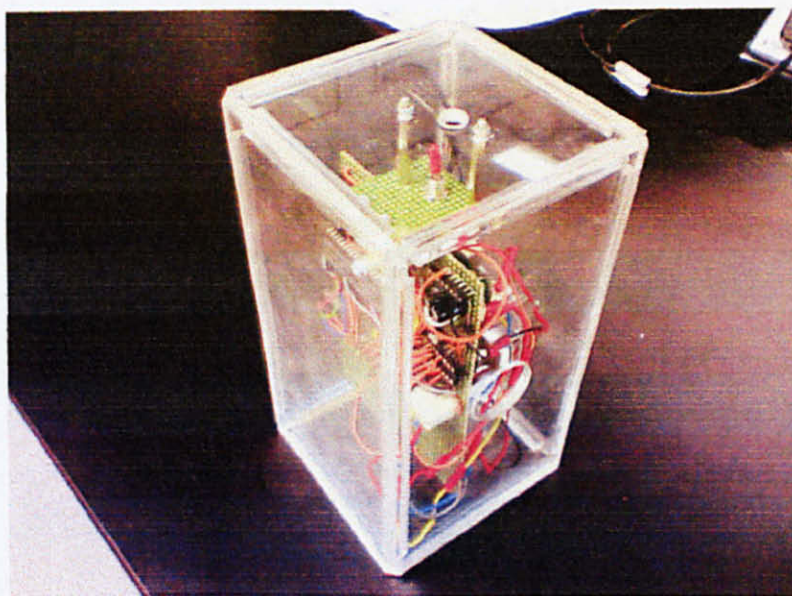
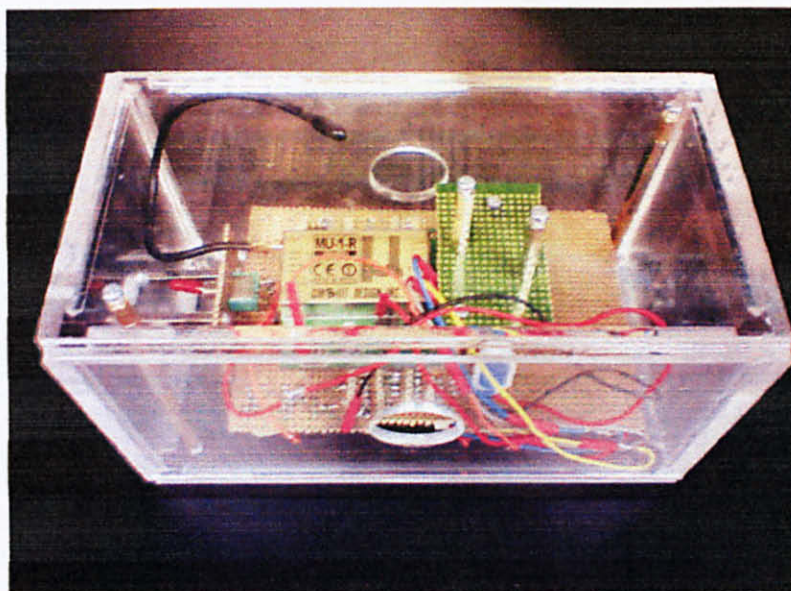
**APPENDIX A**

**FIRST PROTOTYPE OF CANSAT**



## APPENDIX B

### FINAL PROTOTYPE OF CANSAT





## APPENDIX C

### MICROCONTROLLER ALGORITHM FOR CANSAT

```
//this part only up to 1way communication from CanSat to GS
//and CanSat MU-1 response feedback to microcontroller
//and CanSat transmit voltage level of temperature +33decimal
//and CanSat transmit voltage level of X-axis acceleration +33decimal
//and CanSat transmit voltage level of Y-axis acceleration +33decimal
//and CanSat transmit voltage level of Z-axis acceleration +33decimal

#include <stdio.h>
#include <string.h>
#include <pic.h>

__CONFIG (0x3F32);

static unsigned char receivedata20[20] =
{0x00,0x00,0x00,0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};

static unsigned long valueL = 0,sequence = 0,valueLow = 0,valueHi = 0,value = 0;
const char * HELLO = "@DT05HELLO/R01";
const char * RTNA = "@RTNA";
const char * EI02 = "@EI02";
const char * GI02 = "@GI02";
const char * GOOD = "@DT04GOOD/R01";
const char * datastored = 0;

void uartwrite9k6(unsigned char datachar);
void uartwrite19k2(unsigned char datachar);
void uarttransmit(const char * datachar);
```



```

void uartreceive19k2(void);
unsigned char SPI(unsigned char myByte);

void main()
{
    TRISB = 0b11111110;           // Port B bits 7 and 6 are output
    TRISD = 0b00000000;
    TRISA = 0b00011111;
    TRISC = 0b10000000;
    PEIE = 1;
    GIE = 1;
    //RCIE = 1;

    PORTB = 0;
    PORTD = 0;

    for(valueL = 0; valueL < 100000; valueL++);
    uarttransmit(RTNA);
    for(valueL = 0; valueL < 100000; valueL++);
    uarttransmit(EI02);
    for(valueL = 0; valueL < 100000; valueL++);
    uarttransmit(GI02);

    while(1)
    {
        for(valueL = 0; valueL < 100000; valueL++);
        uarttransmit(HELLO);

        for(sequence = 0; sequence < 6; sequence++)
        {
            TXSTA = 0b00000100;
            SPBRG = 0b01000000;

```

```

SPEN = 1;
CREN = 1;
while(RCIF == 0);
    receivedata20[sequence + 5] = RCREG;
}
CREN = 0;

```

```

for(valueL = 0; valueL < 100000; valueL++);
uartwrite19k2('@');
uartwrite19k2('D');
uartwrite19k2('T');
uartwrite19k2('0');
uartwrite19k2('3');
uartwrite19k2(receivedata20[5]);
uartwrite19k2(receivedata20[6]);
uartwrite19k2(receivedata20[7]);
uartwrite19k2('/');
uartwrite19k2('R');
uartwrite19k2('0');
uartwrite19k2('1');
uartwrite19k2(0x0D);
uartwrite19k2(0x0A);

```

```

for(valueL = 0; valueL < 100000; valueL++);
ADCON1 = 0b10000010;
ADCON0 = 0b10100000;
ADON = 1;
ADCON0 |= 0b00000100;
while((ADCON0 & 0x04) == 1);
valueLow = ADRESL;
valueHi = ADRESH;
valueHi = valueHi << 8;

```

```

value = ((valueLow + valueHi)*255)/1023;
receivedata20[1] = value + 33;

uartwrite19k2('@');
uartwrite19k2('D');
uartwrite19k2('T');
uartwrite19k2('0');
uartwrite19k2('4');
uartwrite19k2('T');
uartwrite19k2('C');
uartwrite19k2('=');
uartwrite19k2(receivedata20[1]);
uartwrite19k2('/');
uartwrite19k2('R');
uartwrite19k2('0');
uartwrite19k2('1');
uartwrite19k2(0x0D);
uartwrite19k2(0x0A);

for(valueL = 0; valueL < 100000; valueL++);
ADCON1 = 0b10000010;
ADCON0 = 0b10001000;
ADON = 1;
ADCON0 |= 0b00000100;
while((ADCON0 & 0x04) == 1);
valueLow = ADRESL;
valueHi = ADRESH;
valueHi = valueHi << 8;
value = ((valueLow + valueHi)*255)/1023;
receivedata20[2] = value + 33;

uartwrite19k2('@');

```



```

uartwrite19k2('D');
uartwrite19k2('T');
uartwrite19k2('0');
uartwrite19k2('4');
uartwrite19k2('A');
uartwrite19k2('X');
uartwrite19k2('=');
uartwrite19k2(receivedata20[2]);
uartwrite19k2('/');
uartwrite19k2('R');
uartwrite19k2('0');
uartwrite19k2('1');
uartwrite19k2(0x0D);
uartwrite19k2(0x0A);

for(valueL = 0; valueL < 100000; valueL++);
ADCON1 = 0b10000010;
ADCON0 = 0b10010000;
ADON = 1;
ADCON0 |= 0b00000100;
while((ADCON0 & 0x04) == 1);
valueLow = ADRESL;
valueHi = ADRESH;
valueHi = valueHi << 8;
value = ((valueLow + valueHi)*255)/1023;
receivedata20[3] = value + 33;

uartwrite19k2('@');
uartwrite19k2('D');
uartwrite19k2('T');
uartwrite19k2('0');
uartwrite19k2('4');

```

```

uartwrite19k2('A');
uartwrite19k2('Y');
uartwrite19k2('=');
uartwrite19k2(receivedata20[3]);
uartwrite19k2('/');
uartwrite19k2('R');
uartwrite19k2('0');
uartwrite19k2('1');
uartwrite19k2(0x0D);
uartwrite19k2(0x0A);

for(valueL = 0; valueL < 100000; valueL++);
ADCON1 = 0b10000011;
ADCON0 = 0b10011000;
ADON = 1;
ADCON0 |= 0b00000100;
while((ADCON0 & 0x04) == 1);
valueLow = ADRESL;
valueHi = ADRESH;
valueHi = valueHi << 8;
value = ((valueLow + valueHi)*255)/1023;
receivedata20[4] = value + 33;

uartwrite19k2('@');
uartwrite19k2('D');
uartwrite19k2('T');
uartwrite19k2('0');
uartwrite19k2('4');
uartwrite19k2('A');
uartwrite19k2('Z');
uartwrite19k2('=');
uartwrite19k2(receivedata20[4]);

```

```

        uartwrite19k2('/');
        uartwrite19k2('R');
        uartwrite19k2('0');
        uartwrite19k2('1');
        uartwrite19k2(0x0D);
        uartwrite19k2(0x0A);
    }
}

void uartreceive19k2(void)
{
    for(sequence = 0; sequence < 6; sequence++)
    {
        TXSTA = 0b00000100;
        SPBRG = 0b01000000;
        SPEN = 1;
        CREN = 1;
        while(RCIF == 0);
        receivedata20[sequence] = RCREG;
    }
    CREN = 0;
}

void uarttransmit(const char * datachar)
{
    unsigned char datasequence = 0;
    do
    {
        uartwrite19k2(datachar[datasequence]);
        datasequence++;
        if(datachar[datasequence] == 0)
            datasequence = 21;
    }
    while(datasequence != 21);
}

```



```

    datachar = 0;
    uartwrite19k2(0x0D);
    uartwrite19k2(0x0A);
}

void uartwrite19k2(unsigned char datachar)
{
    TXSTA = 0b00000100;
    SPBRG = 0b01000000;
    SPEN = 1;
    TXEN = 1;
    TXREG = datachar;
    while(TXIF == 0);
    while(TRMT == 0);
}

void uartwrite9k6(unsigned char datachar)
{
    TXSTA = 0b00000100;
    SPBRG = 0b10000010;
    SPEN = 1;
    TXEN = 1;
    TXREG = datachar;
    while(TXIF == 0);
    while(TRMT == 0);
}

```

## APPENDIX D

### VISUAL BASIC 2008 ALGORITHM FOR MISSION CONTROL CENTER (MCC)

```
Imports System
Imports System.Drawing
Imports System.IO.Ports
Imports System.Threading
Imports System.Threading.Thread
Imports System.Drawing.Drawing2D

Public Class MaxiTester
    Dim SpaceCount As Byte = 0
    Dim LookupTable As String = "0123456789ABCDEF"
    Dim RXArray(2047) As Char ' Text buffer. Must be global to be
accessible from more threads.
    Dim RXCnt As Integer      ' Length of text buffer. Must be global
too.
    Dim TextString As String
    Dim RXByte As Byte
    Dim result As Integer
    Dim RXArray2(2047) As Char
    Dim RXCnt2 As Integer = 0
    Dim sample As Integer = 199
    Dim samplebuffery As Integer = 0
    Dim samplebufferx As Integer = 0

    ' Make a new System.IO.Ports.SerialPort instance, which is able to
fire events.
    Dim WithEvents COMPort As New SerialPort

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim I As Integer

        Dim g1 As Graphics = Me.CreateGraphics()
        g1.DrawLine(Pens.Black, 60, 325, 960, 325)
        g1.DrawLine(Pens.Blue, 960, 70, 960, 325)
        For I = 0 To 199
            g1.DrawLine(Pens.Black, (60 + (I * 4)), 325, (60 + (I * 4)),
330)
        Next
```

```

End Sub

Private Sub Receiver(ByVal sender As Object, ByVal e As
SerialDataReceivedEventArgs) Handles COMPort.DataReceived
    Dim RXByte As Byte
    Do
        RXCnt = 0
        Do
            RXByte = COMPort.ReadByte
            RXArray2(RXCnt2) = ChrW(RXByte)
            RXCnt2 = RXCnt2 + 1

        Loop Until (COMPort.BytesToRead = 0)

        '----- End of communication protocol handling -----
        Me.Invoke(New MethodInvoker(AddressOf Display)) ' Start
        "Display" on the UI thread
        Loop Until (COMPort.BytesToRead = 0) ' Don't return if more
        bytes have become available in the meantime
    End Sub

    ' Text display routine, which appends the received string to any
    text in the Received TextBox.

Private Sub Display()
    Received.AppendText(New String(RXArray, 0, RXCnt))
    TextBox1.AppendText(New String(RXArray2, 0, RXCnt2))
    Displayprocessed()
    'RXArray2 = 0
End Sub

' Transmitter subroutine.

Private Sub Send(ByVal sender As Object, ByVal e As EventArgs)
Handles SendButton.Click
    Transmitter()
End Sub

Private Sub Transmitter()
    Received.AppendText("TX" & vbCrLf) ' Switch to a new line
after every transmission
    SpaceCount = 0
    Dim TextString As String
    Dim TXArray(2047) As Byte
    Dim I As Integer
    Dim J As Integer = 0
    Dim Ascii As Boolean = False

```



```

Dim Quote As Boolean = False
Dim Temp As Boolean
Dim Second As Boolean = False
Dim TXByte As Byte = 0
Dim CharByte As Byte
If COMPort.IsOpen Then
    TextString = Transmitted.Text
    For I = 0 To TextString.Length - 1
        CharByte = Asc(TextString.Chars(I))
        If CharByte = 34 Then ' If " Then
            Temp = Ascii
            Ascii = Ascii Or Quote
            Quote = Not (Temp And Quote)
        Else
            Ascii = Ascii Xor Quote
            Quote = False
        End If
        If Not Quote Then
            If Ascii Then
                TXArray(J) = CharByte
                J = J + 1
            Else
                If (CharByte <> 32) And (CharByte <> 10) And
(CharByte <> 13) Then ' Skip spaces, LF and CR
                    CharByte = (CharByte - 48) And 31 ' And 31
makes it case insensitive
                If CharByte > 16 Then
                    CharByte = CharByte - 7
                End If
                If Second Then
                    TXArray(J) = TXByte + CharByte
                    Second = False
                    J = J + 1
                Else
                    TXByte = CharByte << 4
                    Second = True
                End If
            End If
        End If
    Next
    Try
        COMPort.Write(TXArray, 0, J)
    Catch ex As Exception
        MsgBox(ex.Message & " Check CTS signal or set Flow
Control to None.")
    End Try
Else

```

```

        MsgBox("COM port is closed. Please select a COM port")
    End If
End Sub

Private Sub PortSelection(ByVal sender As Object, ByVal e As
EventArgs) Handles COMPortsBox.SelectedIndexChanged
    RTSLamp.BackColor = Color.Gray
    DTRLamp.BackColor = Color.Gray
    If COMPort.IsOpen Then
        COMPort.RtsEnable = False
        COMPort.DtrEnable = False
        ClosePort()
        Application.DoEvents()
        Sleep(200) ' Wait 0.2 second for port
to close as this does not happen immediately.
    End If
    COMPort.PortName = COMPortsBox.Text
    COMPort.BaudRate = 19200 ' Default for Max-i: 19200
bit/s, 8 data bits, no parity, 1 stop bit
    COMPort.WriteTimeout = 2000 ' Max time to wait for CTS =
2 sec.
    ' COMPort.ReadBufferSize = 16384 Necessary buffer size for
16C950 UART at 921.6 kbit/s
    Try
        COMPort.Open()
    Catch ex As Exception
        MsgBox(ex.Message)
    End Try
    BaudRateBox.Text = COMPort.BaudRate.ToString
    BitsBox.Text = COMPort.DataBits.ToString
    ParityBox.Text = COMPort.Parity.ToString
    FlowControlBox.Text = COMPort.Handshake.ToString
    If COMPort.IsOpen Then
        COMPort.RtsEnable = True
        RTSLamp.BackColor = Color.LightGreen
        COMPort.DtrEnable = True
        DTRLamp.BackColor = Color.LightGreen
    End If
End Sub

Private Sub MaxiTesterLoad(ByVal sender As Object, ByVal e As
EventArgs) Handles MyBase.Load
    For Each COMString As String In
My.Computer.Ports.SerialPortNames ' Load all available COM ports.
        COMPortsBox.Items.Add(COMString)
    Next
    COMPortsBox.Sorted = True
    BaudRateBox.Items.Add("110")

```

```

BaudRateBox.Items.Add("300")
BaudRateBox.Items.Add("600")
BaudRateBox.Items.Add("1200")
BaudRateBox.Items.Add("1800")
BaudRateBox.Items.Add("2400")
BaudRateBox.Items.Add("4800")
BaudRateBox.Items.Add("7200")
BaudRateBox.Items.Add("9600")
BaudRateBox.Items.Add("14400")
BaudRateBox.Items.Add("19200")      ' Min. FIFO size 3 Bytes
(9030 or 8530)
BaudRateBox.Items.Add("38400")
BaudRateBox.Items.Add("57600")      ' Min. FIFO size 8 bytes
BaudRateBox.Items.Add("115200")      ' Min. FIFO size 16 bytes
(16C550)
BaudRateBox.Items.Add("230400")      ' Min. FIFO size 32 bytes
(16C650)
BaudRateBox.Items.Add("460800")      ' Min. FIFO size 64 bytes
(16C750)
BaudRateBox.Items.Add("921600")      ' Min. FIFO size 128 bytes
(16C850 or 16C950)
BitsBox.Items.Add("5")
BitsBox.Items.Add("6")
BitsBox.Items.Add("7")
BitsBox.Items.Add("8")
ParityBox.Items.Add("None")
ParityBox.Items.Add("Odd")
ParityBox.Items.Add("Even")
ParityBox.Items.Add("Mark")          ' Leaves the parity bit set
to 1
ParityBox.Items.Add("Space")          ' Leaves the parity bit set
to 0
FlowControlBox.Items.Add("None")
FlowControlBox.Items.Add("RequestToSend")
FlowControlBox.Items.Add("RequestToSendXOnXOff")
FlowControlBox.Items.Add("XOnXOff")
RTSLamp.BackColor = Color.Gray
DTRLamp.BackColor = Color.Gray
CTSLamp.BackColor = Color.Gray
DSRLamp.BackColor = Color.Gray
End Sub

Private Sub ClosePort()
    If COMPort.IsOpen Then COMPort.Close()
End Sub

Private Sub MaxiTesterClosing(ByVal sender As Object, ByVal e As
ComponentModel.CancelEventArgs) Handles MyBase.Closing

```



```

        If MessageBox.Show("Do you really want to close the window", "",
        MessageBoxButtons.YesNo) = Windows.Forms.DialogResult.No Then
            e.Cancel = True
        Else
            ' Close COM port on a new thread when the form is terminated
with [X]
            Dim t As New Thread(AddressOf ClosePort)
            t.Start()
        End If
    End Sub

    Private Sub ClearReceivedText(ByVal sender As Object, ByVal e As
EventArgs) Handles ClearButton.Click
        Received.Text = ""
        SpaceCount = 0
    End Sub

    Private Sub SendBreak(ByVal sender As Object, ByVal e As EventArgs)
Handles BreakButton.Click
        If COMPort.IsOpen Then
            COMPort.BreakState = True
            Sleep((11000 / COMPort.BaudRate) + 10)
            ' Min. 11 bit delay (startbit, 8 data bits, parity bit,
stopbit). 10 mS have been added to ensure that
            ' the delay is always active. If the delay time is less
than the task switching time, the two BreakState
            ' instructions may be activated immediately after each
other ! On a multiprocessor system, you must
            ' add 15 mS instead.
            COMPort.BreakState = False
        Else
            MsgBox("No COM Port Selected")
        End If
    End Sub

    Private Sub BaudRateSelection(ByVal sender As Object, ByVal e As
EventArgs) Handles BaudRateBox.SelectedIndexChanged
        COMPort.BaudRate = CInt(BaudRateBox.Text)
    End Sub

    Private Sub DataBitsSelection(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles BitsBox.SelectedIndexChanged
        COMPort.DataBits = CInt(BitsBox.Text)
    End Sub

    Private Sub ParitySelection(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ParityBox.SelectedIndexChanged

```

```

        COMPort.Parity = CType([Enum].Parse(GetType(Parity),
ParityBox.Text), Parity)
    End Sub

    Private Sub SoftwareFlowControlSelection(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
FlowControlBox.SelectedIndexChanged
        COMPort.Handshake = CType([Enum].Parse(GetType(Handshake),
FlowControlBox.Text), Handshake)
    End Sub

    Private Sub ModemLamps(ByVal sender As Object, ByVal e As
SerialPinChangedEventArgs) Handles COMPort.PinChanged
        If COMPort.DsrHolding Then
            DSRLamp.BackColor = Color.LightGreen
        Else
            DSRLamp.BackColor = Color.Gray
        End If
        If COMPort.CtsHolding Then
            CTSLamp.BackColor = Color.LightGreen
        Else
            CTSLamp.BackColor = Color.Gray
        End If
    End Sub

    Private Sub SaveText(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        Dim SaveFileDialog1 As New SaveFileDialog()
        SaveFileDialog1.Filter = "Text Files (*.txt)|*.txt"
        SaveFileDialog1.Title = "Save Received As"
        If SaveFileDialog1.ShowDialog() =
System.Windows.Forms.DialogResult.OK _
            And SaveFileDialog1.FileName.Length > 0 Then
            My.Computer.FileSystem.WriteAllText(SaveFileDialog1.FileName,
Received.Text, False) ' Overwrite file
        End If
    End Sub

    Private Sub Transmitted_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Transmitted.TextChanged

    End Sub

    Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click
        Transmitted.Text = "40 47 49 30 32 0D 0A"
        Transmitter()
    End Sub

```

```

End Sub

Protected Overrides Sub OnPaint(ByVal e As PaintEventArgs)
    Dim I As Integer

    Dim g1 As Graphics = Me.CreateGraphics()
    g1.DrawLine(Pens.Black, 60, 325, 960, 325)
    g1.DrawLine(Pens.Blue, 960, 70, 960, 325)
    For I = 0 To 199
        g1.DrawLine(Pens.Black, (60 + (I * 4)), 325, (60 + (I * 4)),
330)
    Next
End Sub 'OnPaint

'This is the Paint event handler
Private Sub Form1_Paint(ByVal sender As Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles MyBase.Paint

End Sub 'Form1_Paint

Private Sub Received_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Received.TextChanged

End Sub

Private Sub ReceivedLabel_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles ReceivedLabel.Click

End Sub

Private Sub Displayprocessed()
    Dim I As Integer

    Dim g1 As Graphics = Me.CreateGraphics()

    'g1.DrawLine(Pens.Black, (60 + (I * 4)), 325, (60 + (I * 4)),
330)
    sample = sample - 1

    For I = 0 To RXArray2.Length - 1

        If RXArray2(I) = "T" Then
            If RXArray2(I + 1) = "C" Then
                Temp.AppendText(AscW(RXArray2(I + 3)) & vbCrLf)
                If sample = 198 Then
                    g1.DrawLine(Pens.Red, 60 + (sample * 4), 325 -
(AscW(RXArray2(I + 3))), 60 + (sample * 4), 325 - (AscW(RXArray2(I +
3))))

```



```

        samplebufferx = 60 + (sample * 4)
        samplebuffery = 325 - AscW(RXArray2(I + 3))
    Else
        g1.DrawLine(Pens.Red, 60 + (sample * 4), 325 -
(AscW(RXArray2(I + 3))), samplebufferx, samplebuffery)
        samplebufferx = 60 + (sample * 4)
        samplebuffery = 325 - AscW(RXArray2(I + 3))
    End If
End If
End If
If RXArray2(I) = "A" Then
    If RXArray2(I + 1) = "X" Then
        Xaxis.AppendText(AscW(RXArray2(I + 3)) & vbCrLf)
    End If
End If
If RXArray2(I) = "A" Then
    If RXArray2(I + 1) = "Y" Then
        Yaxis.AppendText(AscW(RXArray2(I + 3)) & vbCrLf)
    End If
End If
If RXArray2(I) = "A" Then
    If RXArray2(I + 1) = "Z" Then
        Zaxis.AppendText(AscW(RXArray2(I + 3)) & vbCrLf)
    End If
End If
Next

If sample = 0 Then
    sample = 199
End If
End Sub

Private Sub Temp_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Temp.TextChanged

End Sub

Private Sub TextBox1_TextChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles TextBox1.TextChanged

End Sub

Private Sub Xaxis_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Xaxis.TextChanged

End Sub

```

```
Private Sub Yaxis_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Yaxis.TextChanged

End Sub

Private Sub Zaxis_TextChanged(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles Zaxis.TextChanged

End Sub

End Class
```