EDUCAN MINI MANUAL

ORION SPACE PVT. LTD.
KATHMANDU, NEPAL

Table of Contents

Introduction

A CanSat is a simulation of a real satellite, integrated within the volume and shape of a soft drink can. It is a type of Sounding Rocket Payload used to teach space technology.It is based on similar technology used in Miniaturized Satellites.The challenge is to fit all the major subsystems found in a satellite, such as power, sensors and a communication system, into this minimal volume somewhere between 300-350gm. The CanSat is then launched to an altitude of a few hundred meters by a rocket or dropped from a platform or captive balloon and its mission begins: to carry out a scientific experiment and achieve a safe landing.

CanSat offer a unique opportunity to have first practical experience of a real space project. They are responsible for all aspects: designing the CanSat, selecting its mission, integrating the components, testing, preparing for launch and then analyzing the data.

CanSat is for everyone interested in electronics and Space Technology. Our CanSat consist of a payload, ESP8266 board as the On Board Computer as well as Communication and a charging unit for the battery installed within the CanSat.

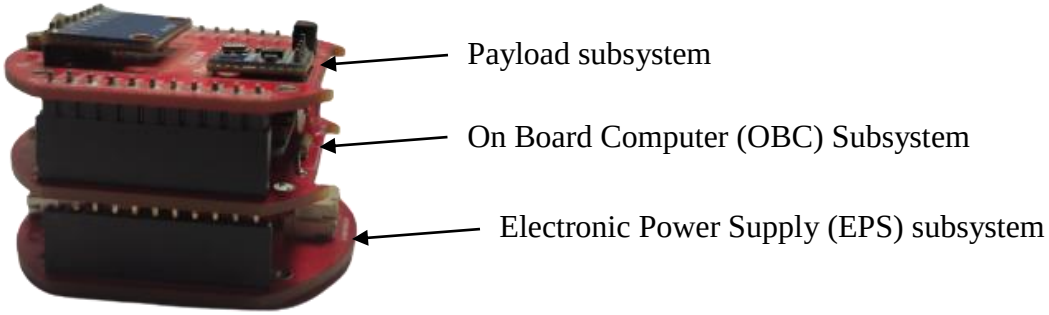


Fig: EDUCAN-MINI

Features of CanSat

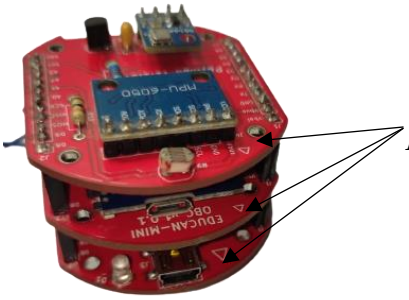
- Very short time is required to complete a CanSat project.
- 5-6 months for mission conceptualization, satellite design, fabrication, ground test, modification, launch, and operation with a variety of hands-on
- Very low life-cycle cost for one project
- It is small in size but the majority of satellite features are included
- It can be retrieved after an experiment
- It’s just a simulation of a real satellite so no need to launch into space
- It’s a high possibility to get sponsored by a juice or cola company.

EDUCan-MINI Specification:

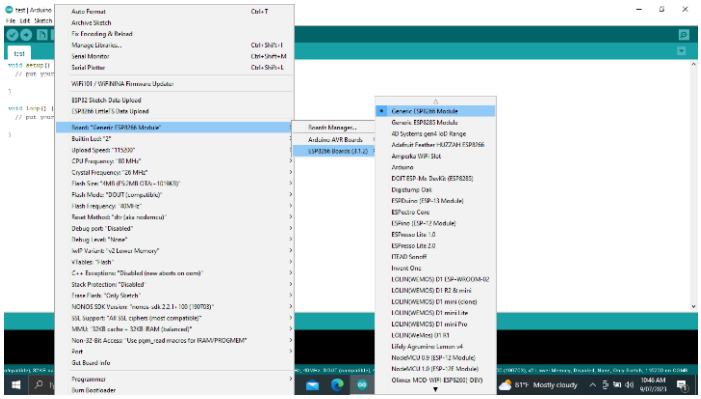
- This EDUCan-MINI is 3 Layered.
- The bottom layer is the EPS subsystem which supplies power to the whole system.
- Second Layer is On Board Computer acts like the brain of the whole system along with it have Wi-Fi module embedded so also act like a communication subsystem.
- Top layer is the payload subsystem and contains various sensors for data acquisition

Some Warnings

- Check the Polarity of Battery while Connecting.
- While Stacking the board on top of each other see the arrow or name of pins.



- For the Arduino IDE choose Board Generic ESP8266 Module.



- To upload program , remove all other subsystem stacked from OBC.
- For fully functional of the CANSAT, connect OBC , Payload and EPS subsystem before operating it.
- Connect the host with the AP of the ESP8266 before trying to receiving data to the IP address

What are in the CanSat EduCAN Box?

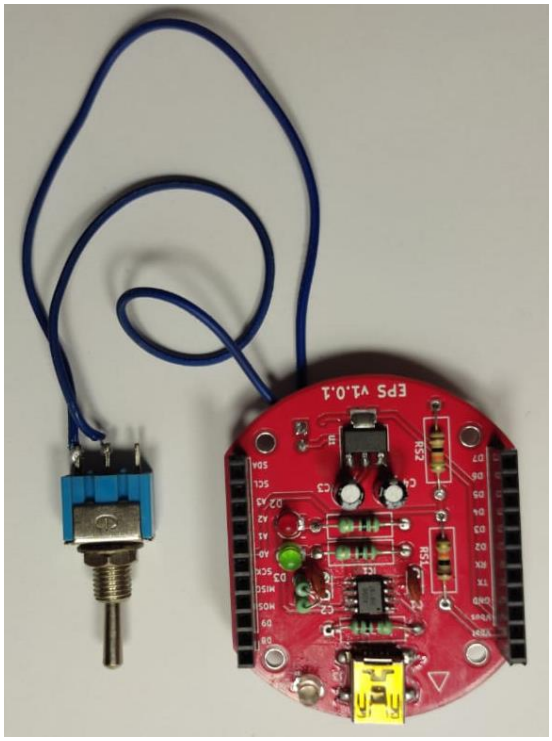
A CanSat EduCAN box is a CanSat Educational Kit designed for basic learning and understanding of CanSat. This EduCan Box outlines the main features of the Primary Mission for CanSat. In the Primary Mission, teams must measure the temperature and pressure and send the information to their ground station. You will learn about the differences between the sensors they can use and about the challenges associated with completing the Mission. It is designed in line with a range of resources to support the entire CanSat mission.

The CanSat EduCan Box consists of

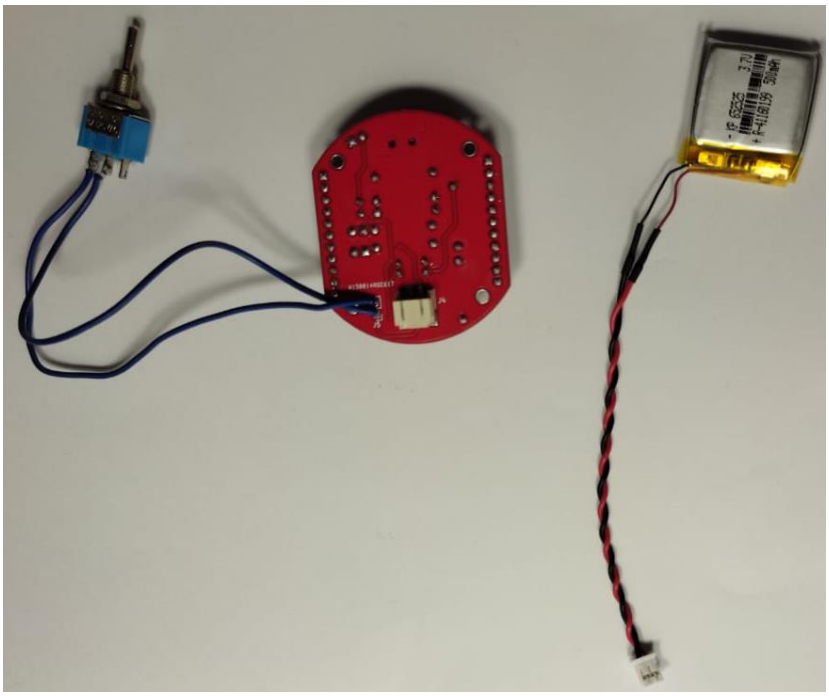
1. 3D Printed Can Shape with mounted holes.
2. ESP8266 processor (OBC) *1
3. Payload Board *1
4. Electronic Power Supply Board *1
5. Li-Po Battery *1
6. Male Header Pins
7. Female Header Pins
8. Toggle Switch
9. LDR
10. MPU6050 3-axis Accelerometer & Gyroscope Sensor
11. BMP180 Pressure Sensor
12. LM35 Temperature Sensor
13. Battery Connector Wire
14. Resistors
15. LED
16. Screws
17. Brass Rods

CANSAT ASSEMBLY

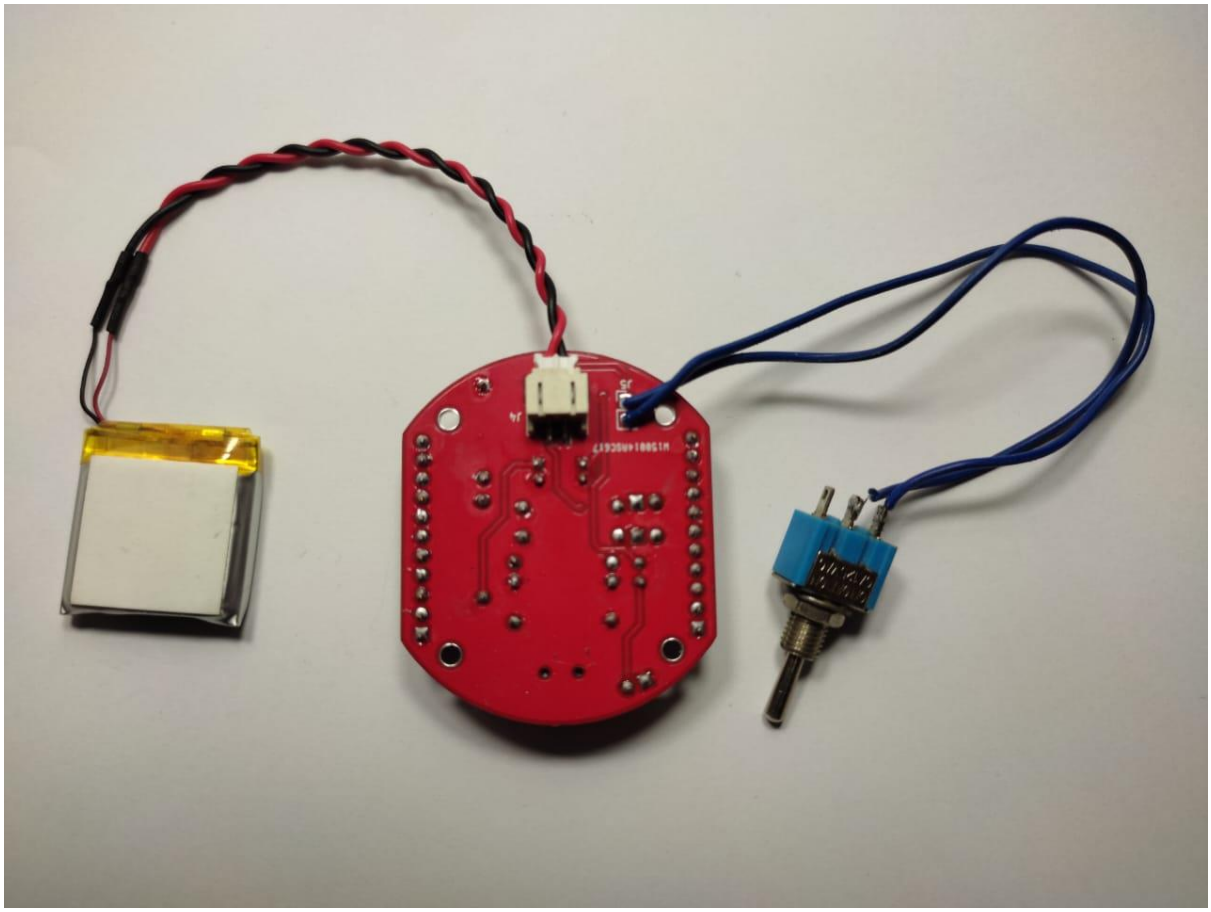
1. Take the EPS subsystem. Your EPS subsystem should look exactly like this.

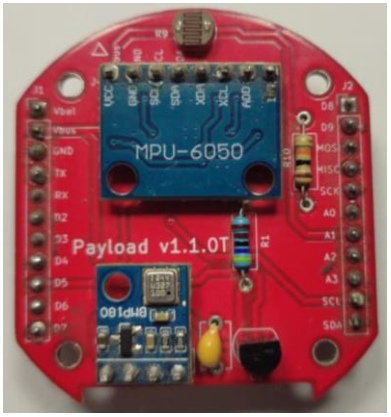


2. Take the Battery provided. After completion of soldering of the EPS. The connector will connect battery and EPS.



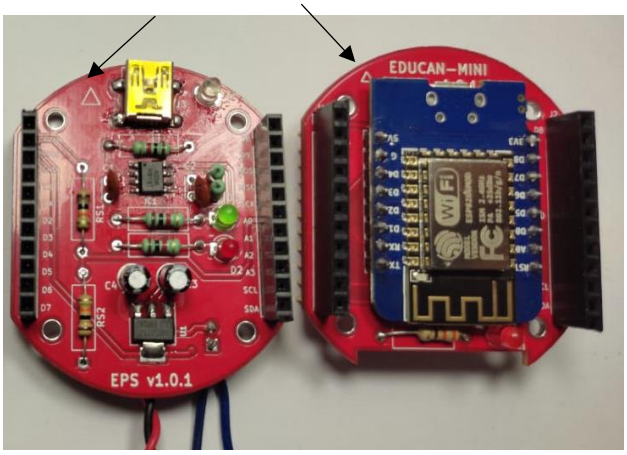
3. The final EPS sub system looks like this. Note that after you on the toggle switch, The LED of EPS glows. This indicating the EPS is successfully assembled.

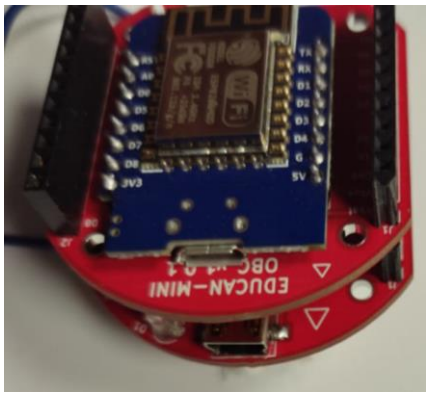




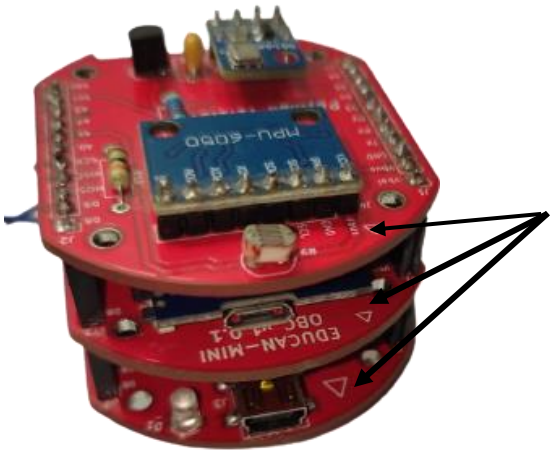
4. The payload subsystem should exactly look like this. After successful assembling of the Payload, EPS and OBC subsystem, now it's time to assemble all subsystem together.

Note the arrow facing outward in same direction





5. Now after the stack of both subsystem correctly, it's time to stack the payload above it.
6. The earlier notation of arrow is also marked in payload. Align these in accordance to the OBC subsystem.



Note arrow facing outward in all subsystem.

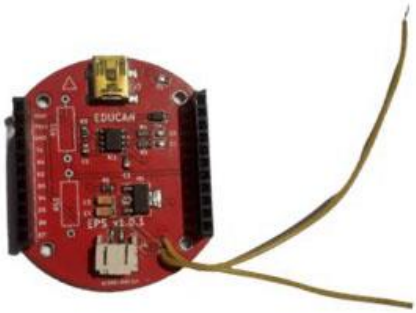
7. Now Place all component inside the box of CANSAT so that it become ready to launch.

CanSat Kit:

The CanSat Kit consists of:

- Electronic Power Supply (EPS) Subsystem
- Onboard Computer (OBC) Subsystem
- Payload Subsystem

Electronic Power Supply (EPS) Subsystem



Electronics Power Supply (EPS) Subsystem is the board responsible For the power distribution to all other subsystems of the CanSat.

This portion consists of a connection for battery and current and Voltage sensors for effective power distribution across the system.

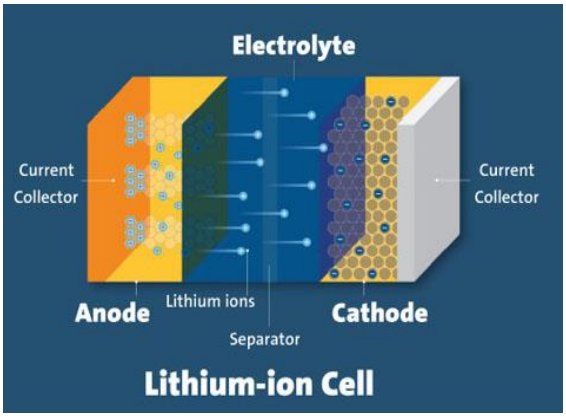
Li-ion battery

- Li-ion battery is a rechargeable advanced battery technology that uses lithium (Li) ion.
- It is the most suitable battery for CanSat power supply

Li-ion 1000mAH Battery

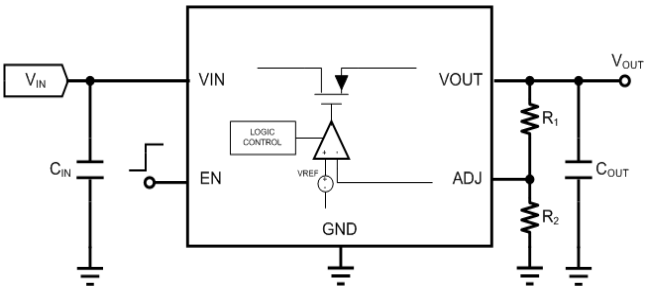
Features:

- It is composed of cells in which lithium ions move from the negative electrode through an electrolyte
- To the positive electrode during discharge and back when charging.
- Energy density: 250–693 W·h/L(0.90–2.43 MJ/L)
- Specific energy: 100–265 W·h/kg(0.36–0.875 MJ/kg)
- Charge/discharge efficiency: 80–90%
- Cycle durability: 400–1,200 cycles
- Energy/consumer-price: 7.6Wh/US$; US $132/kWh
- Self-discharge rate: 0.35% to 2.5% per month depending
- on the state of charge
- Specific power: ~250 – ~340 W/kg



Low-dropout regulator (LDO)

- It is a DC linear voltage regulator that can regulate the output voltage even when the supply voltage is very close to the output voltage.
- Components: power FET and differential amplifier (error amplifier)
- The output voltage is given by:
- $V_{OUT} = \left(1 + \frac{R_1}{R_2}\right) V_{REF}$
- The advantages of an LDO regulator over other DC-to-DC voltage regulators include the absence of switching noise, smaller device size, and greater design simplicity.
- The disadvantage is that, unlike switching regulators, linear DC regulators must dissipate power, and thus heat, across the regulation device in order to regulate the output voltage.



TP4056 Module

- With the inclusion of a battery in any system a battery charger is required.
- TP4056 is a low-cost Lithium Ion battery charger controller IC.
- It supports a constant current – constant voltage charging mechanism for a single cell Li-Ion Battery.
- It is available in an 8-pin SOP package and requires very minimum external components in order to build a
- Lithium-Ion battery charger circuit.
- Applications: Mobile phones, GPS devices, Digital camera, USB chargers, etc.

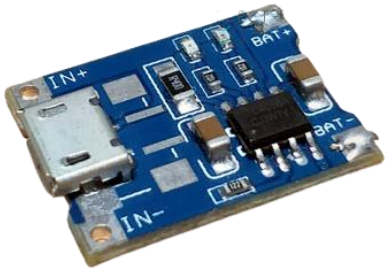
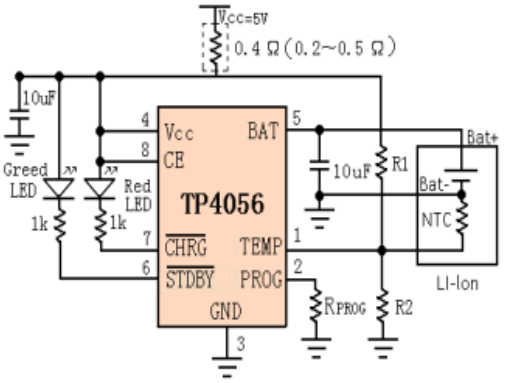


Fig:TP4056module

- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8
- Package for Single Cell Lithium-Ion Batteries
- ·Constant-Current/Constant-Voltage
- ·Charges Single Cell Li-Ion Batteries Directly from USB Port
- ·Preset 4.2V Charge Voltage with 1.5% Accuracy
- ·Automatic Recharge
- ·two Charge Status Output Pins
- ·C/10 Charge Termination
- ·2.9V Trickle Charge Threshold (TP4056)
- Input Supply Voltage(VCC) : -0.3V～8V



On Board Computer (OBC) Subsystem.

On Board Computer (OBC) subsystem

- On Board Computer or OBC is like the brain of a satellite.
- It is responsible for the implementation of control law, processing associated with payload, data packeting activities associated with communication, monitoring load health status, handling of data storage, etc.
- The processor needs to interface with various sensors, actuators present onboard to acquire data to perform its activities, and responds accordingly through actuators.
- Scheduling the activities of the processor is essential due to the number of tasks it has to perform.
- In CanSat microcontrollers like Atmega 328P , ESP8266 , ESP 12S etc. can be used
- We have used ESP8266 as our OBC Subsystem.
- ESP8266 have Wi-Fi module, so we are using it as an AP (Access Point), and hereby this OBC is also acting as Communication Subsystem.
- Choosing of ESP8266 for communication to decrease the hardware requirement of separate communication subsystem, Ground – station and thus reducing the pricing.

ESP8266 specifications

ESP8266

- Microcontroller: Tensilica 32-bit RISC CPU Xtensa LX106
- Integrated low power 32-bit MCU
- Integrated 10-bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA, power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- WiFi 2.4 GHz, support WPA/WPA2
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IR Remote Control, PWM, GPIO
- STBC, 1x1 MIMO, 2x1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4s guard interval
- Deep sleep power <10uA, Power down leakage current < 5uA
- Wake up and transmit packets in < 2ms
- Standby power consumption of < 1.0mW (DTIM3)
- +20 dBm output power in 802.11b mode
- Operating temperature range -40C ~ 125C
- FCC, CE, TELEC, WiFi Alliance, and SRRC certified
- Operating Voltage: 3.3V
- Input Voltage: 7-12V
- Digital I/O Pins (DIO): 16

- Analog Input Pins (ADC): 1
- UARTs: 1
- SPIs: 1
- I2Cs: 1
- Flash Memory: 4 MB
- SRAM: 64 KB
- Clock Speed: 80 MHz
- USB-TTL based on CP2102 is included onboard, Enabling Plug n Play
- PCB Antenna

ESP8266 as Communication Subsystem

Since, ESP8266 has embedded Wi-Fi module, for study purpose and short range, it can also be used as communication subsystem's sender. The receiver is a host browser where it publish the data.

Here is a specification list incorporating the provided data:

• Certificates: FCC, CE, TELEC, SRRC

• Wi-Fi Protocols: 802.11 b/g/n

• Frequency Range: 2.4G-2.5G (2400M-2483.5M)

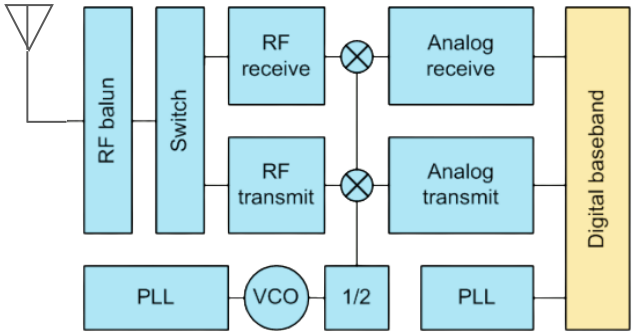
• TX Power:

- 802.11 b: +20 dBm
- 802.11 g: +17 dBm
- 802.11 n: +14 dBm

• Rx Sensitivity:

- 802.11 b: -91 dBm (11 Mbps)
- 802.11 g: -75 dBm (54 Mbps)
- 802.11 n: -72 dBm (MCS7)

• Types of Antenna: PCB Trace, External, IPEX Connector, Ceramic Chip



Block Diagram of ESP8266 Wi-Fi MODULE

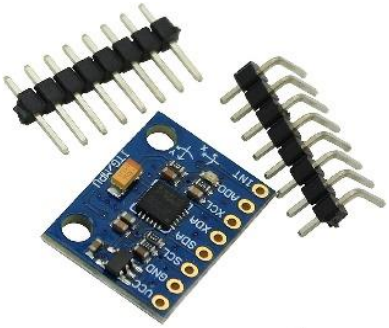
Payload Subsystem

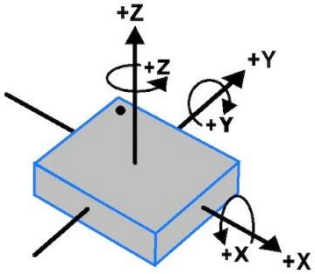
- Payload consist mainly temperature sensor (LM35), pressure sensor (BMP180) and MPU6050 as 3-axis accelerometer and Gyro sensor.
- It also consists of LDR for measurement of intensity of light.

PayLoad SubSystem

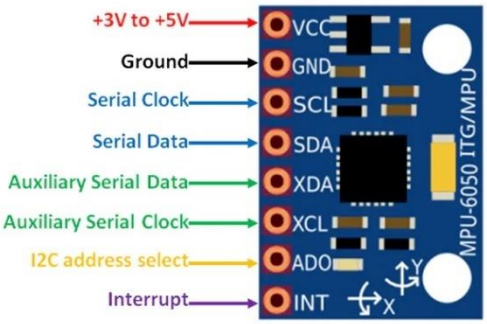
MPU6050

- The MPU6050 consist of 3-axis Gyroscope and 3-axis accelerometer with Micro Electro Mechanical System (MEMS) technology. It helps us to measure velocity, orientation, acceleration, displacement and other motion like features.
- It is used to detect rotational velocity along the X, Y, Z axes.
- Detect angle of tilt or inclination along the X, Y and Z axes.





MPU-6050
Orientation & Polarity of Rotation

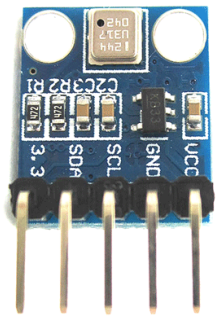


Why use Mpu6050 module?

- MPU6050 consists of Digital Motion Processor (DMP), which has property to solve complex calculations.
- MPU6050 consists of a 16-bit analog to digital converter hardware. Due to this feature, it captures three-dimension motion at the same time.
- This module has some famous features which are easily accessible, due to its easy availability it can be used with a famous microcontroller like Arduino. Friend if you are looking for a sensor to control a motion of your Drone, Self Balancing Robot, RC Cars and something like this, then MPU6050 will be a good choice for you.
- This module uses the I2C module for interfacing with Arduino.

- MPU6050 is less expensive, Its main feature is that it can easily combine with accelerometer and gyro.

BMP180



- Barometric pressure sensor
- BMP180 sensor senses air pressure and provides that information in digital output.
- Can measure temperature and altitude.
- Pressure sensing range: 300-1100 hPa (9000m to -500m above sea level)
- Up to 0.03hPa / 0.25m resolution
- -40 to +85°C operational range, +-2°C temperature accuracy
- High relative accuracy of ±0.12hPa
- Can work on low voltages
- 3.4Mhz I2C interface
- Low power consumption (3uA)
- Pressure conversion time: 5msec
- Potable size

Pin Name	Description
VCC	Connected to +5V
GND	Connected to ground.
SDA	Serial Data pin (I2C interface)
SCL	Serial Clock pin (I2C interface)
3.3V	If +5V is not present. Can power module by connecting +3.3V to this pin.

Why Use BMP180 Module?

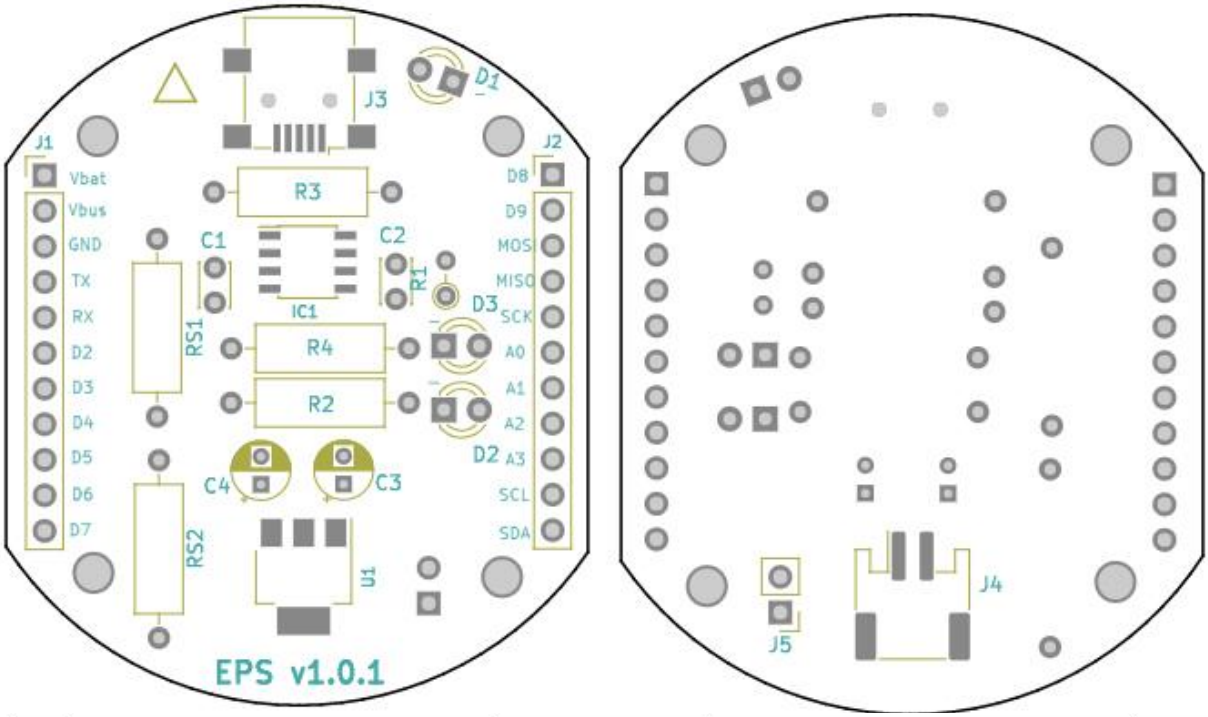
- Accurate Atmospheric Pressure over many modules present in market.
- Consumes less power to function.
- Capable of Communicating with high-speed TWI Interfaces

DS18B20 Temperature sensor

- LM35 is a temperature measuring device having an analog output voltage proportional to the temperature.
- It provides output voltage in Centigrade (Celsius). It does not require any external calibration circuitry.
- The sensitivity of LM35 is 10 mV/degree Celsius. As temperature increases, output voltage also increases.
- E.g.: 250 mV means 25°C.
- It is a 3-terminal sensor used to measure surrounding temperature ranging from -55 °C to 150 °C.
- LM35 gives temperature output which is more precise than thermistor output.

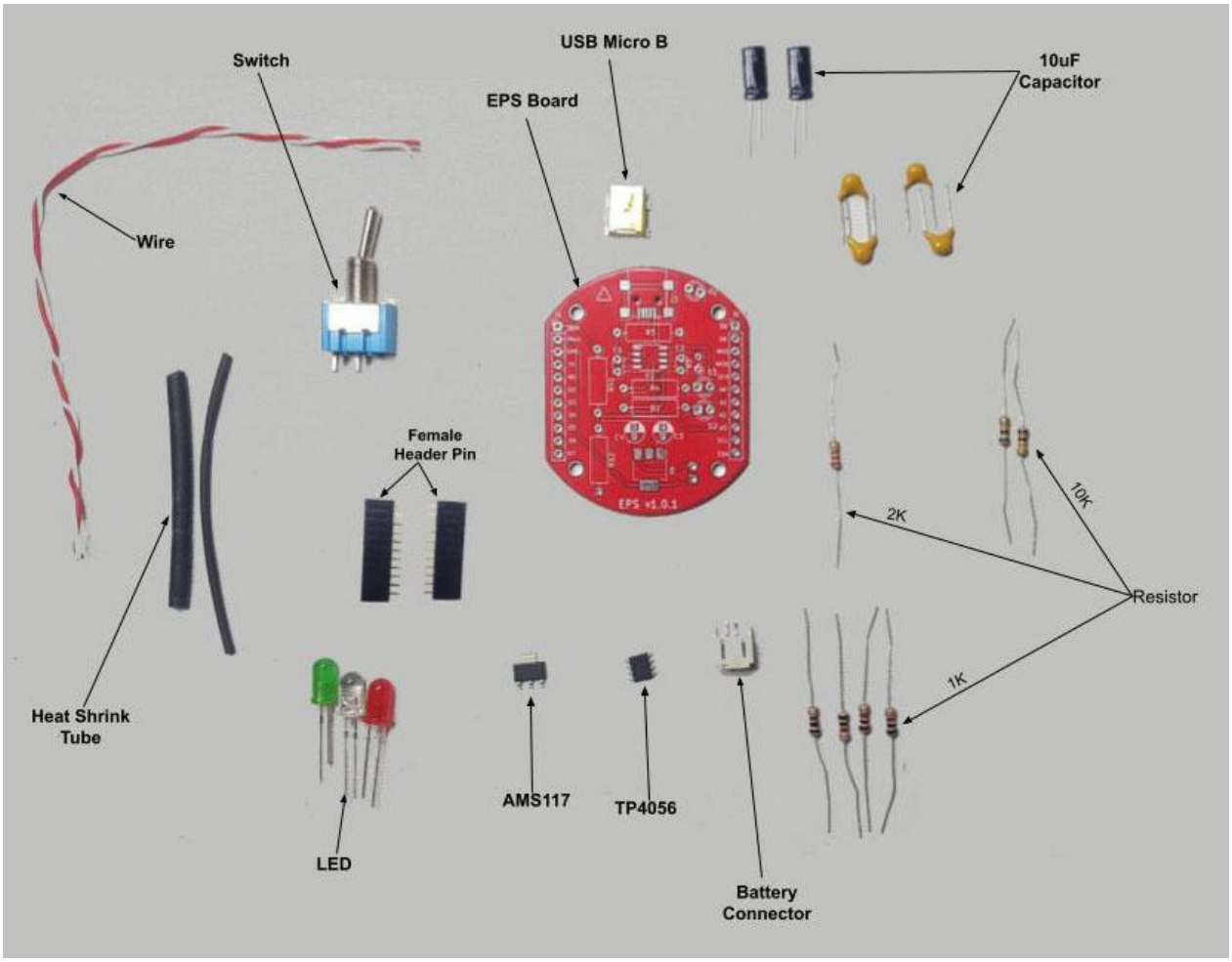
Soldering and Assembly of Subsystems.

Soldering EPS Board



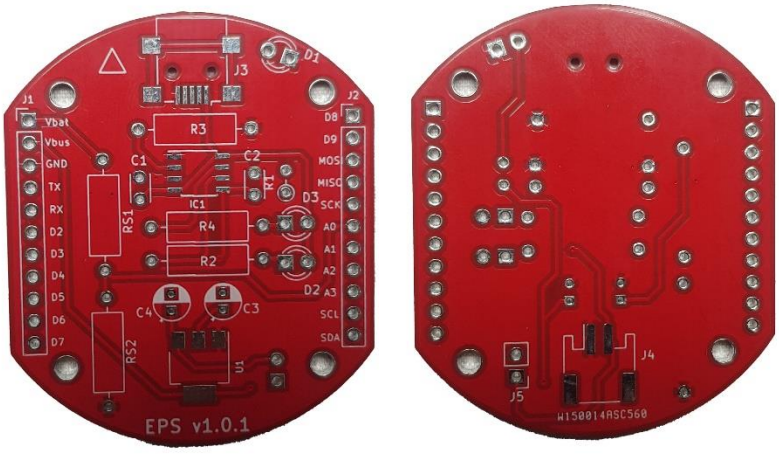
	References	Value	Footprint	Quantity
1	C3, C4	10uf	CP_Radial_D4.0mm_P2.00mm	2
2	C1, C2	10uf	C_Disc_D3.0mm_W2.0mm_P2.50mm	2
3	R1	2k	R_Axial_DIN0204_L3.6mm_D1.6mm_P2.54mm_Vertical	1
4	R2, R4	1k	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	2
5	R3	1.2k	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	1
6	D1	LED	LED_D3.0mm	1
7	D2	GREEN LED	LED_D3.0mm	1
8	D3	RED LED	LED_D3.0mm	1
9	U1	AMS1117-3.3	SOT-223-3_TabPin2	1
10	RS1, RS2	10K	R_Axial_DIN0309_L9.0mm_D3.2mm_P12.70mm_Horizontal	2
11	IC1	TP4056	SOIC-8_3.9x4.9mm_P1.27mm	1
12	J4	CON	JST_PH_S2B-PH-SM4-TB_1x02-1MP_P2.00mm_Horizontal	1
13	J5	Dep-Switch	PinHeader_1x02_P2.54mm_Vertical	1
14	J1	LEFT	PinHeader_1x11_P2.54mm_Vertical	1
15	J2	RIGHT	PinHeader_1x11_P2.54mm_Vertical	1
16	J3	USB_B_Micro	USB_Mini-B_Lumberg_2486_01_Horizontal	1

Step 1: Gather all the components required for the EPS (Electronic Power System) Board.

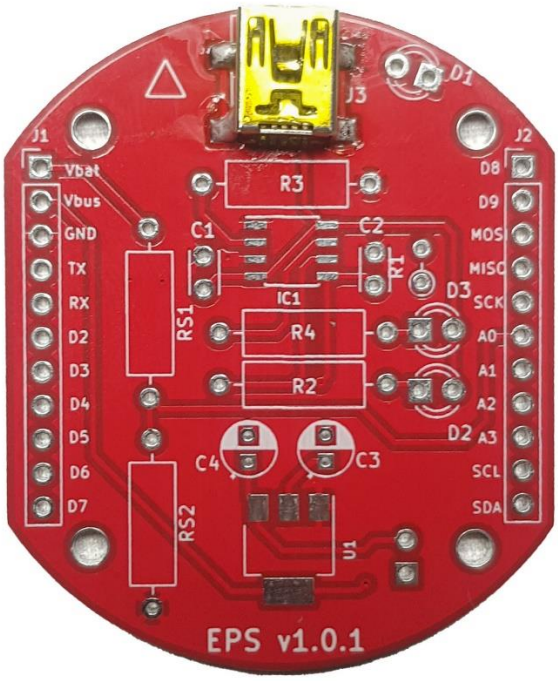


Step 2: See where all the components are to be placed in the EPS Board.

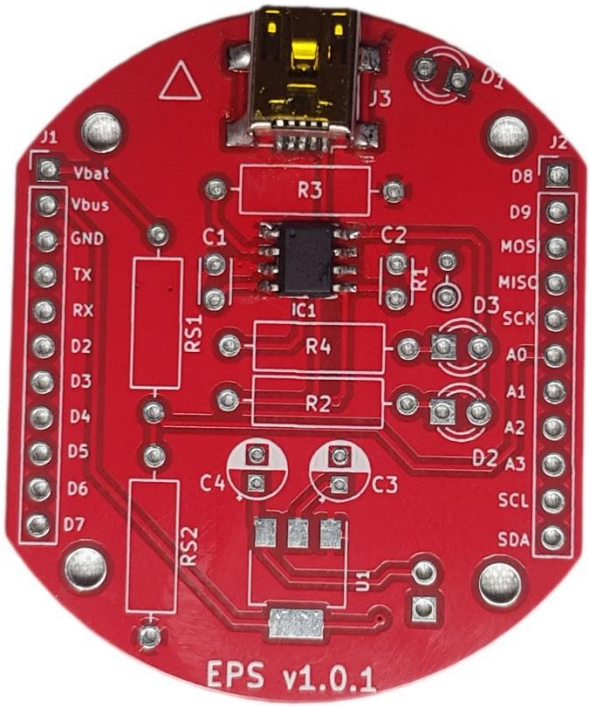
RS1 & RS2- 10K Resistors
R1- 2K Resistor
R2,R3&R4- 1K Resistors
IC1- TP4056
U1- AMS117
J3- USB Micro B
D1- White LED
D2- Green LED
D3- Red LED
C1&C2 – 10uF Ceramic Capacitor
C3&C4 – 10uF Capacitor
J1&J2- Female Header Pins
J4- Battery Connector
J5- Switch Connecting Wire

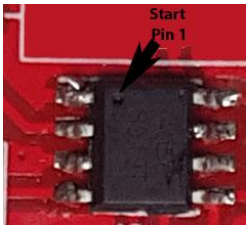


Step 3: Place the USB Micro B in J3 of EPS Board and Solder it to the board.

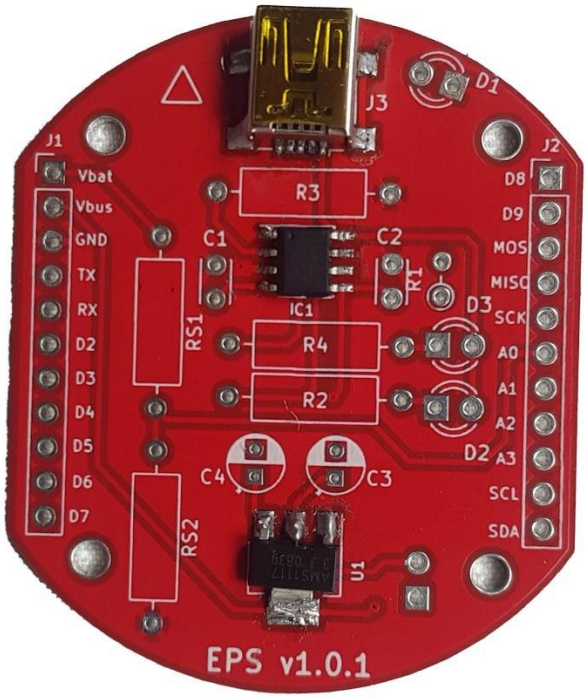


Step 4: Place the TP4056 in IC1 with the circle facing upwards in EPS Board and Solder it to the board.

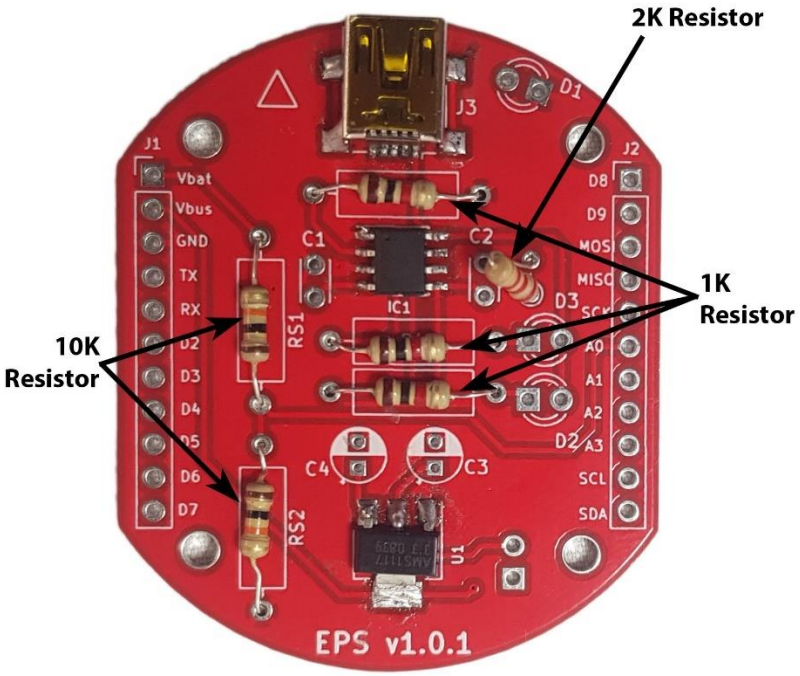




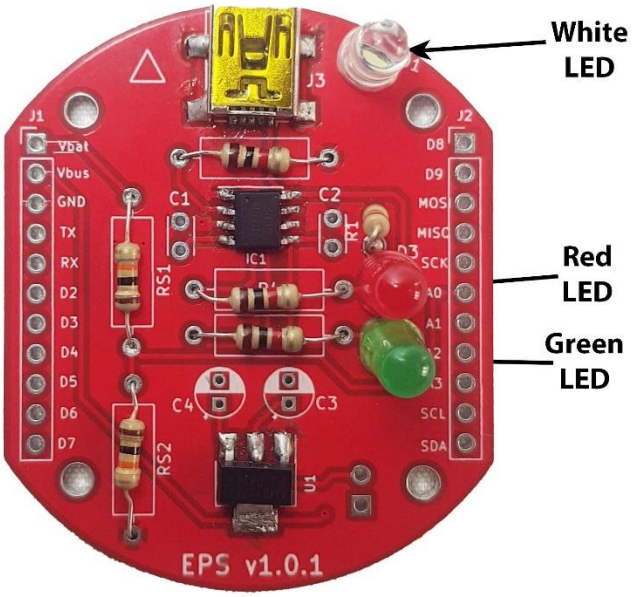
Step 5: Place the AMS117 in U1 in EPS Board and Solder it to the board.

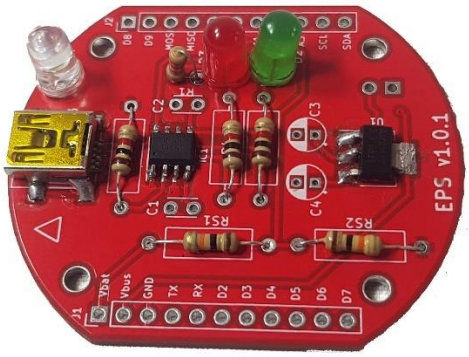


Step 6: Place the 2K resistor in R1,1K resistors in R2,R3&R4 & 10K resistors in RS1 and RS2 in EPS Board and Solder it to the board. Cut off extra wire after soldering.

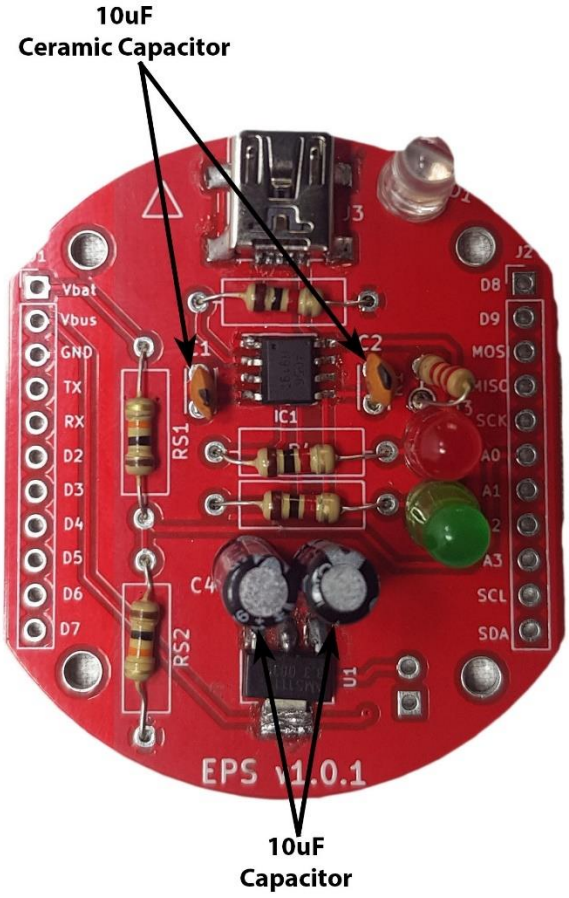


Step 7: Place the White Led in D1,Green Led in D2& Red Led in D3 in EPS Board and Solder it to the board. Cut off extra wire after soldering.

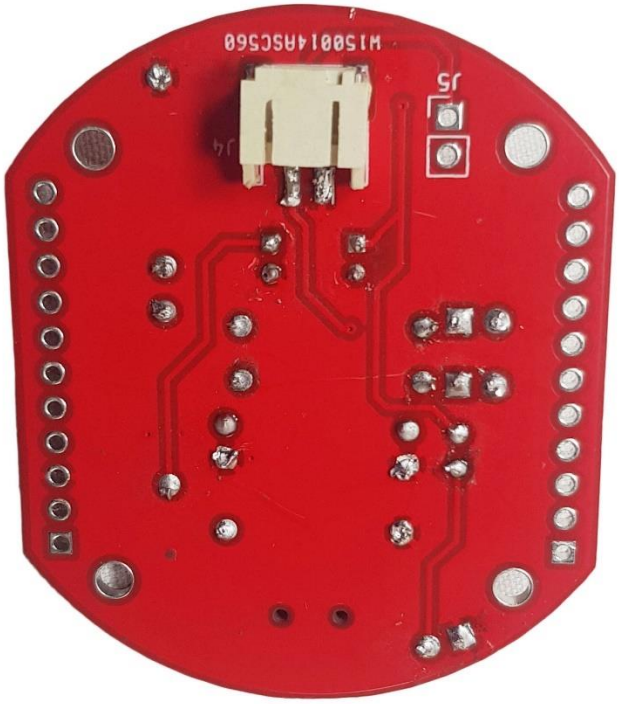




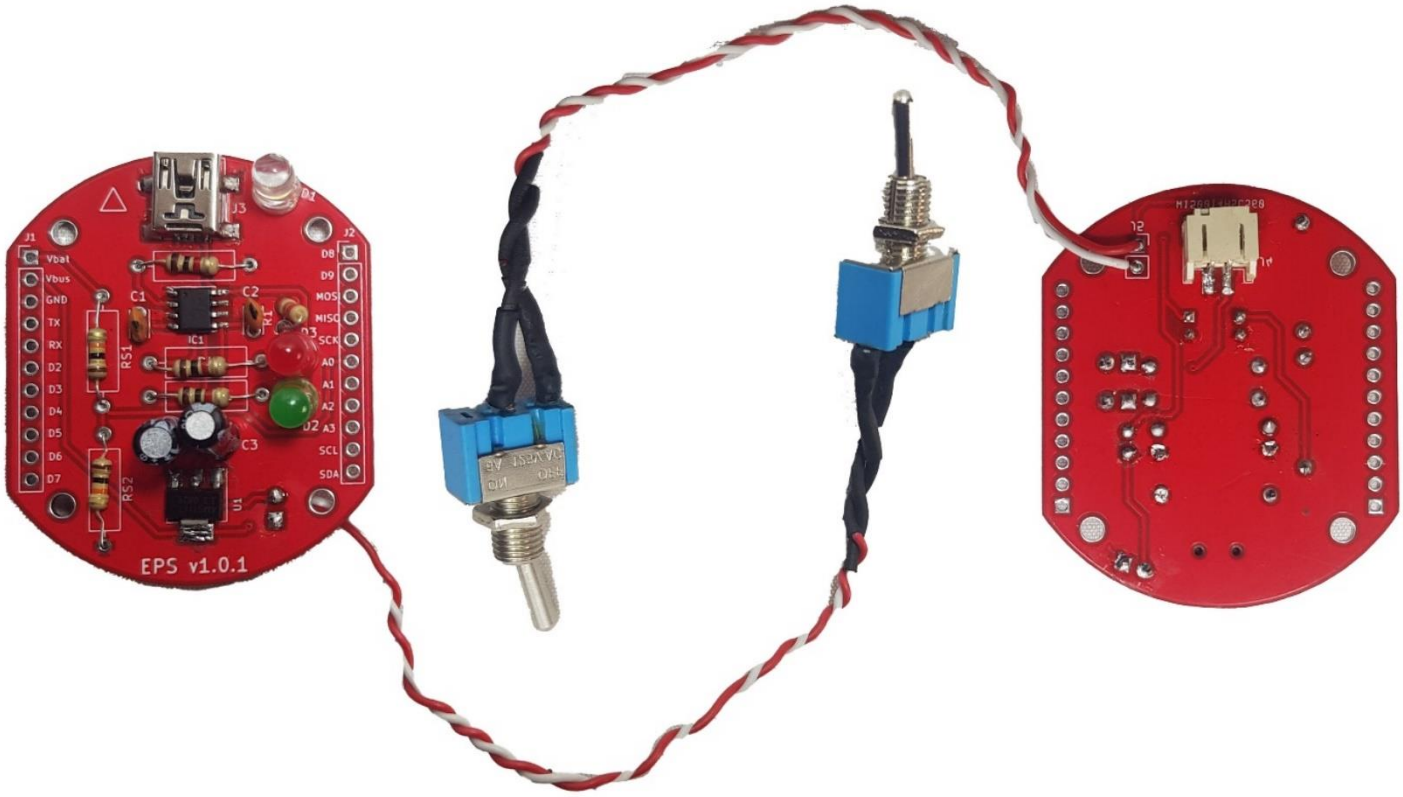
Step 8: Place the 10uF Ceramic Capacitor in C1&C2 and 10uF Capacitor in C3&C4 in EPS Board and Solder it to the board. Cut off extra wire after soldering.



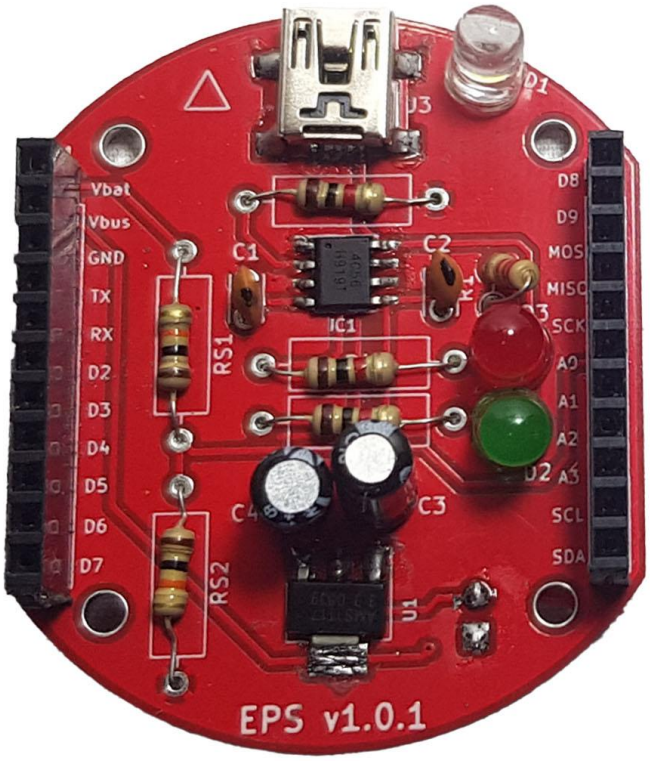
Step 9: Flip the EPS Board and Place the Battery Connector in J4 of EPS Board and Solder it to the board.

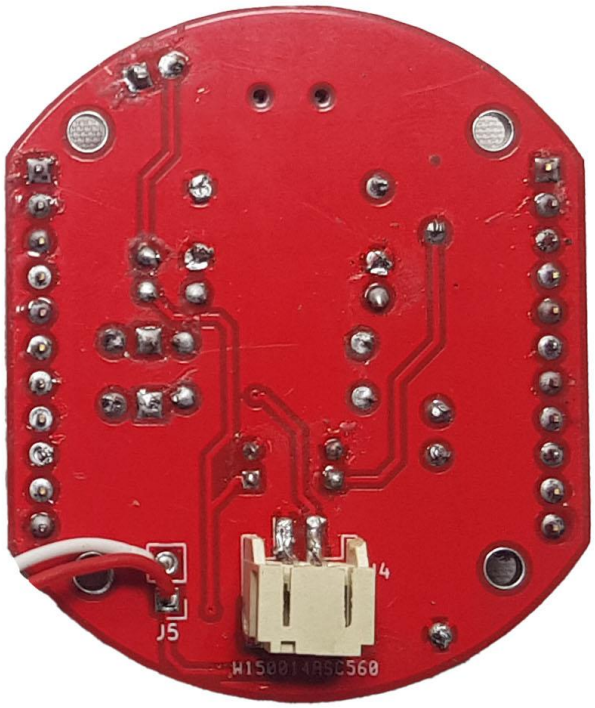


Step 10: Flip the EPS Board and Place the Switch wire in J5 of EPS Board and Solder it to the board. Cut off extra wire after soldering.

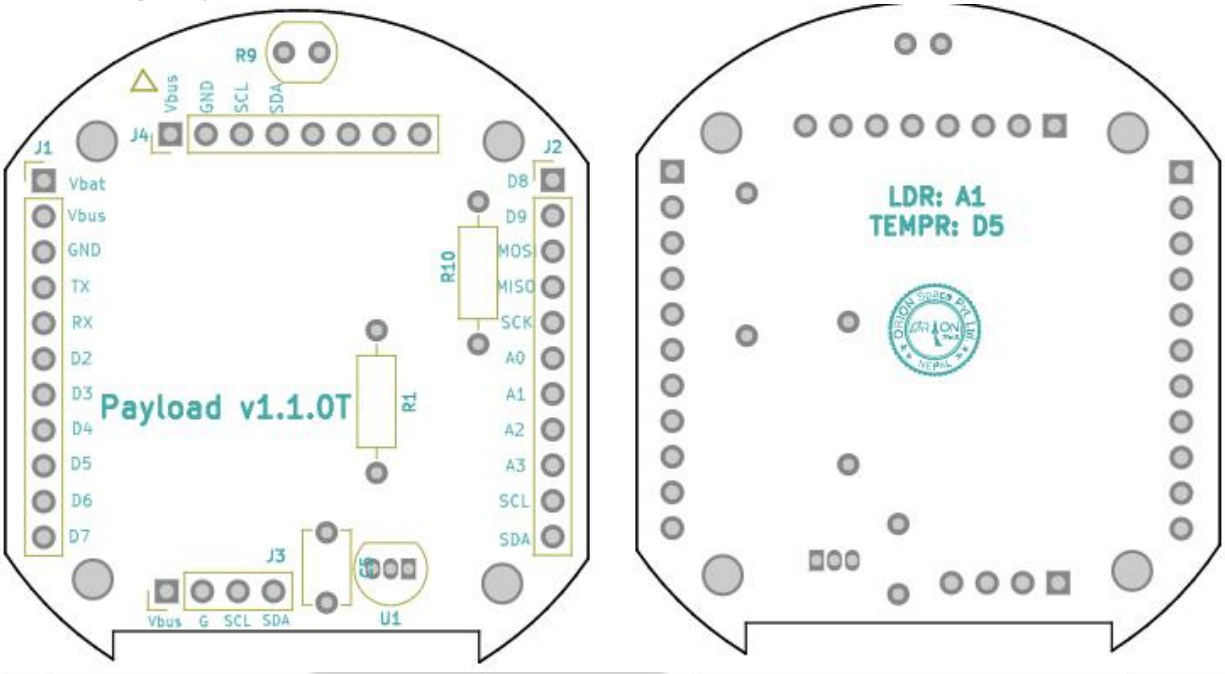


Step 11: Solder the female header pins to the J1&J2 of EPS Board.



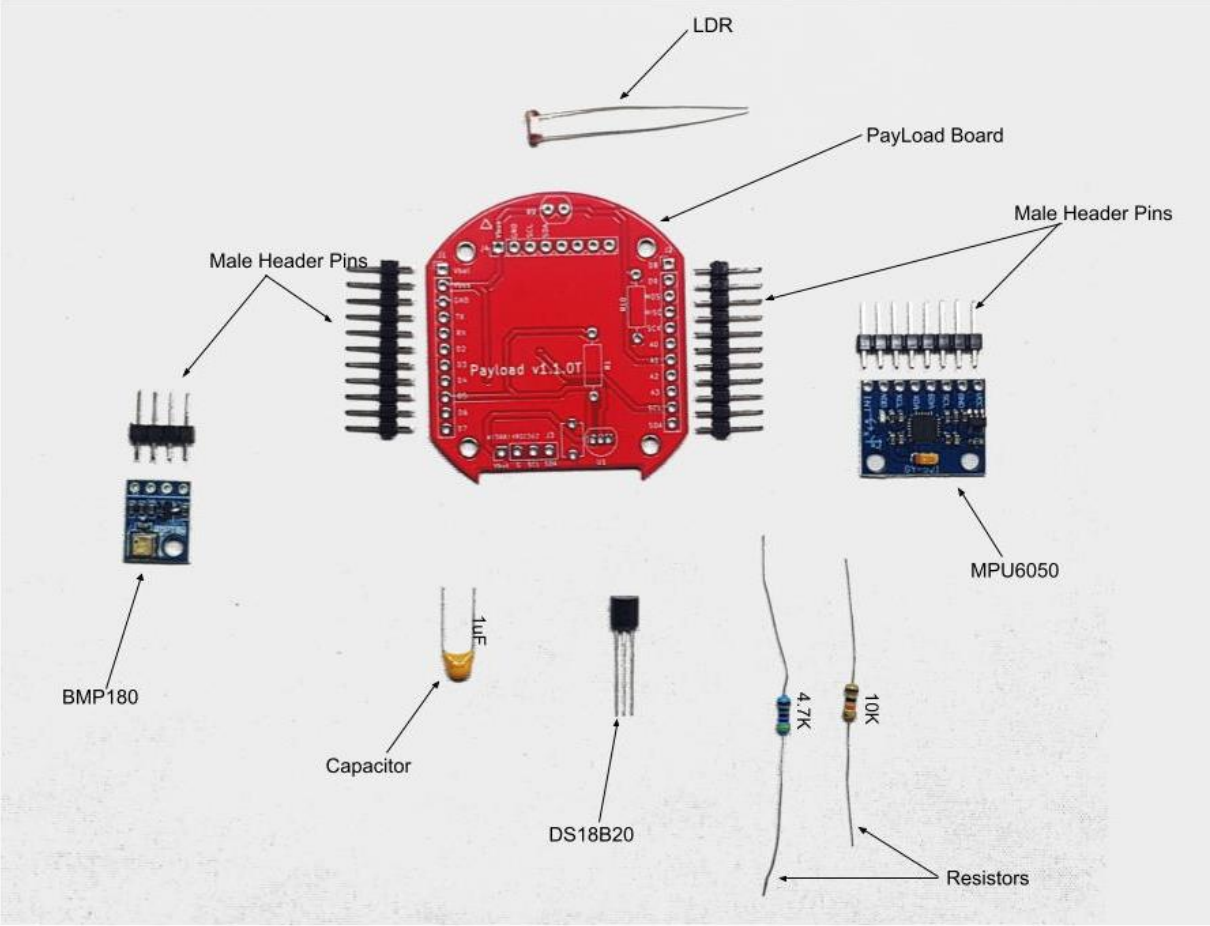


Soldering Payload Board



	References	Value	Footprint	Quantity
1	C5	1uf	C_Disc_D5.1mm_W3.2mm_P5.00mm	1
2	R1, R10	4.7k	R_Axial_DIN0207_L6.3mm_D2.5mm_P10.16mm_Horizontal	2
3	R9	LDR03	R_LDR_4.9x4.2mm_P2.54mm_Vertical	1
4	U1	DS18B20	TO-92_Inline	1
5	G***	LOGO	orion-space-logo_small2	1
6	J3	BMP180	PinHeader_1x04_P2.54mm_Vertical	1
7	J4	MPU6050	PinHeader_1x08_P2.54mm_Vertical	1
8	J1	LEFT	PinHeader_1x11_P2.54mm_Vertical	1
9	J2	RIGHT	PinHeader_1x11_P2.54mm_Vertical	1

Step 1: Gather all the components required for the Payload Board.



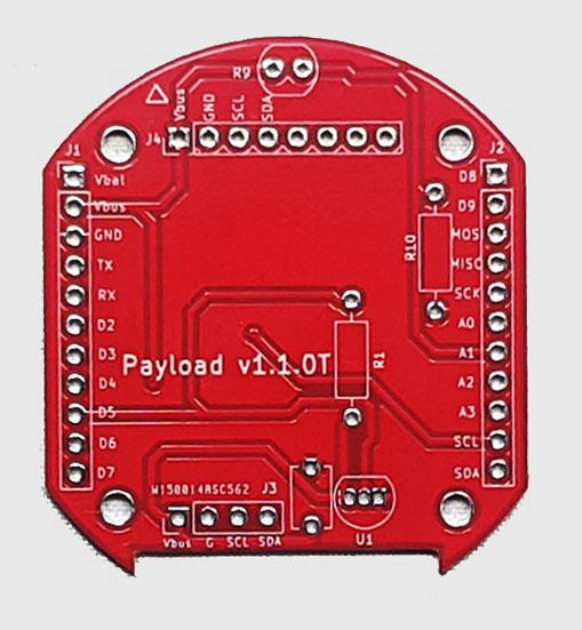
Step 2: See where all the components are to be placed in the Payload Board

J1: Male Header Pins Connector
J2: Male Header Pins Connector
J3: BMP180 Connector
J4: MPU6050 Connector

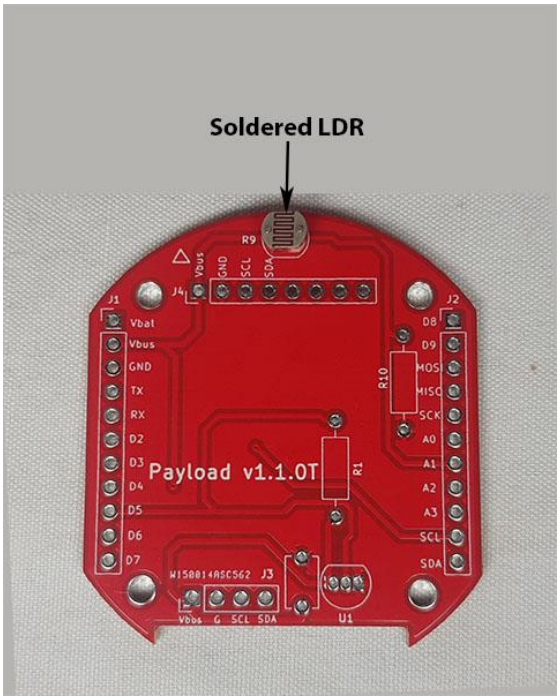
R1 :4.7K Resistor
R10 :10K Resistor

U1: DS18B20 Sensor
R9: LDR Sensor

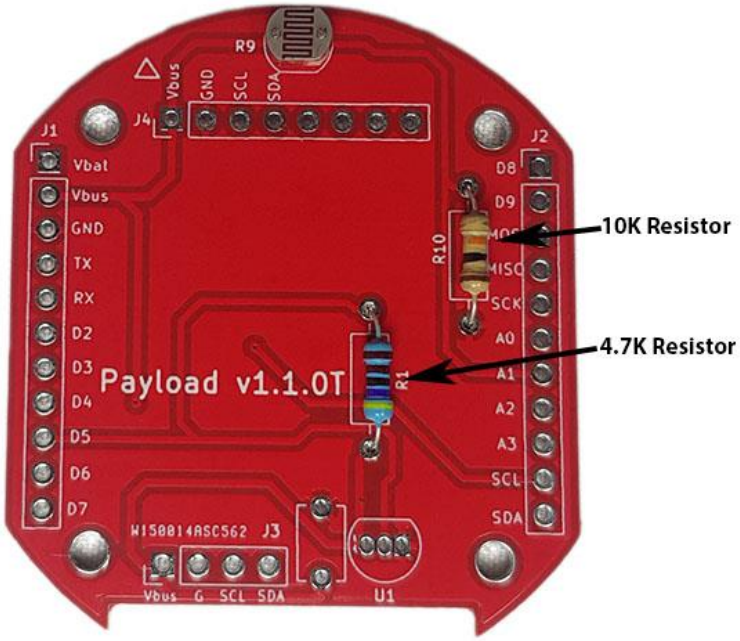
Between J3 & U1: 10uF Capacitor



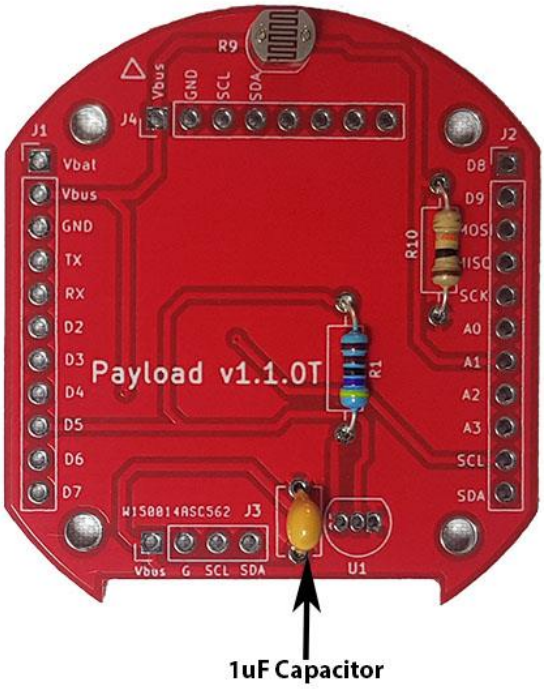
Step 3: Place the LDR sensor in R9 of Payload Board and Solder it to the board. Cut of excess wires after soldering.



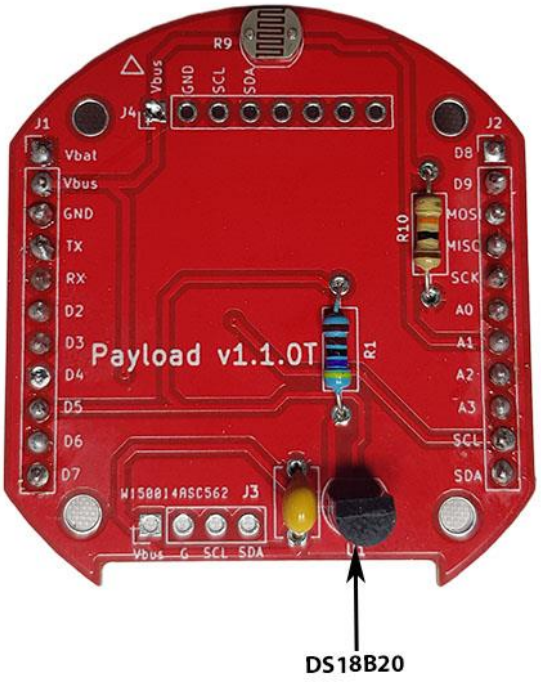
Step 4: Place the 4.7K & 10K resistors in R1 &R10 respectively of Payload Board and Solder it to the board. Cut of excess wires after soldering.



Step 5: Place the 1uF Ceramic Capacitor between J3 & U1 of Payload Board and Solder it to the board. Cut of excess wires after soldering.

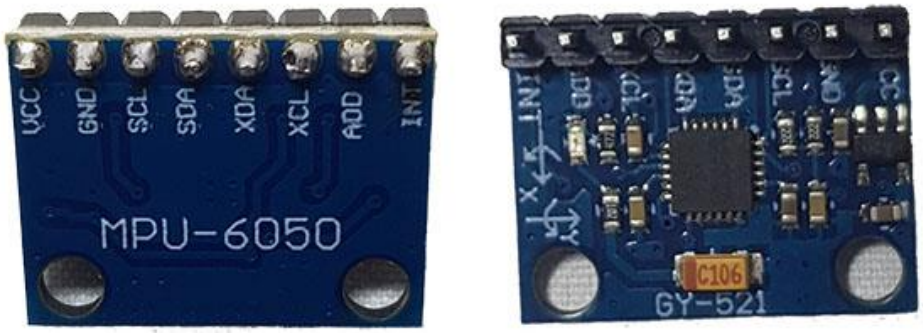


Step 5: Place the DS18B20 Temperature Sensor in U1 of Payload Board and Solder it to the board. Cut of excess wires after soldering.

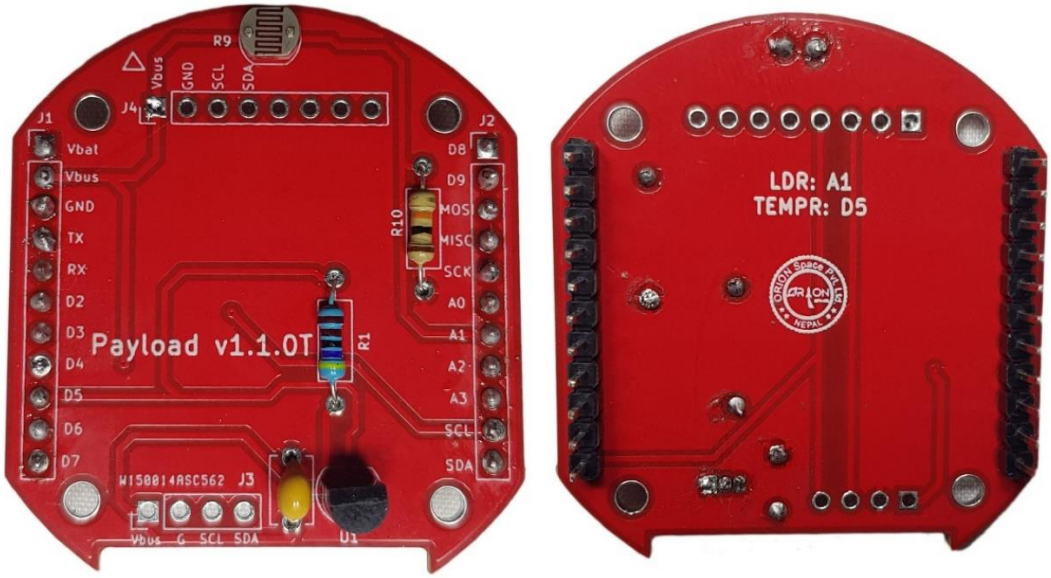


Step 6: Solder the male header pins to the BMP180 Sensor.

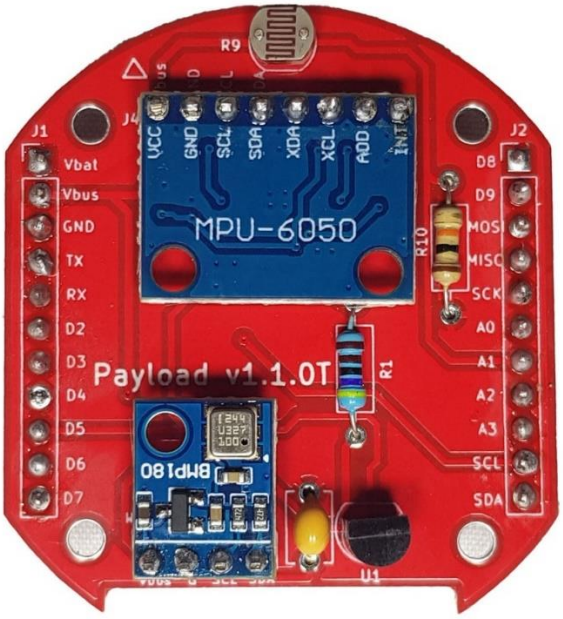
Step 7: Solder the male header pins to the MPU6050 Sensor.



Step 7: Solder the male header pins to the J1&J2 of Payload Board.

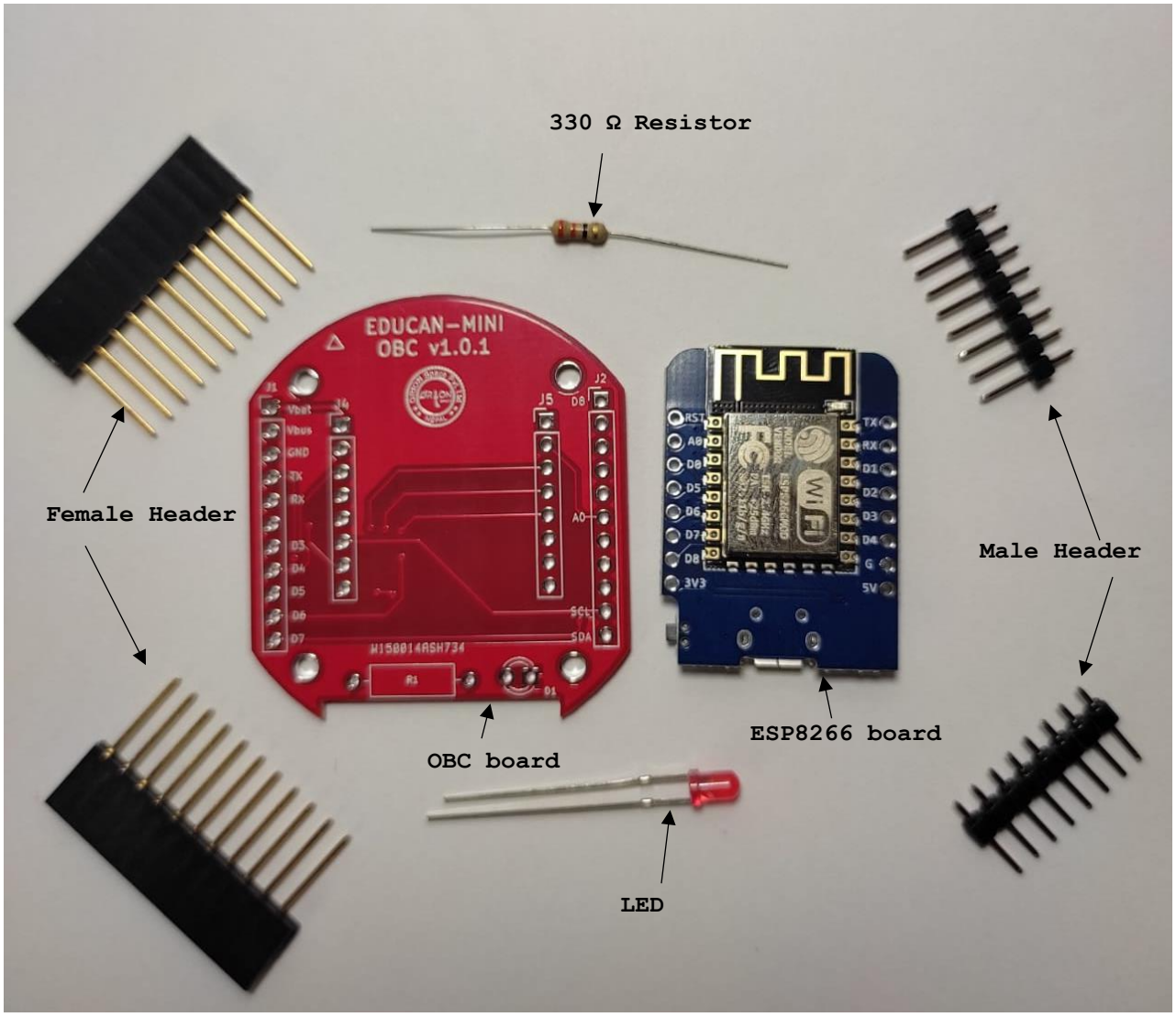


Step 8: Check the pin names printed in the payload board, then attach the MPU6050 & BMP180 sensors in J4 & J3 respectively and solder them.

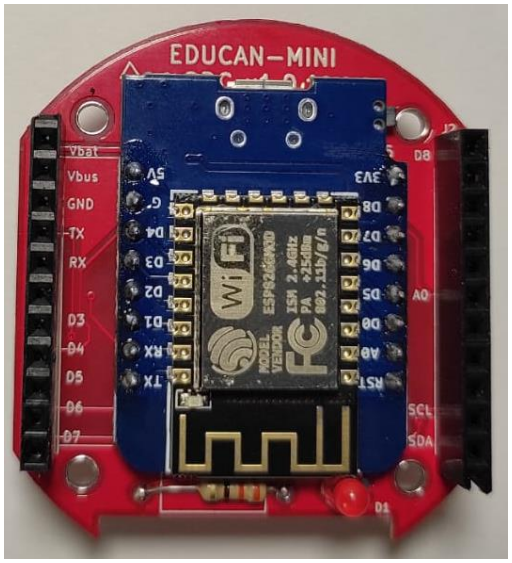


Assembly of OBC subsystem.

1. Gather all the components for OBC board. The components are shown in the figure below.



2. Assemble individual components
    a. At first solder the male header with ESP8266.
    b. Orient the ESP8266 module's 5V with Vbat of OBC board so that the micro USB port of ESP8266 and arrow of EDUCAN-MINI OBC board align in same direction.
    c. Solder male header pins of ESP8266 with the OBC board. They will align perfectly in J4 and J5 soldering trace.
    d. Solder female header pins in outward part of board.
    e. Solder the LED with Cathode(Short legged) part oriented with a square trace and anode in circular trace of D1
    f. Solder the resistor in R1 and trim extra part of both led and resistor.
3. After assembly the board will look exactly like this .



ESP8266 & Arduino IDE

The CanSat has to measure barometric air pressure, temperature, LDR, Gyroscopic and Accelerometer values. The sensors that will be used must be connected to an ESP8266 board. The ESP8266 board can process the sensor signals. The data ESP8266 produces is sent by radio signals with help of the Wi-Fi module Capable of producing an AP of its own. Communication with a ground station (client web server) should be possible.

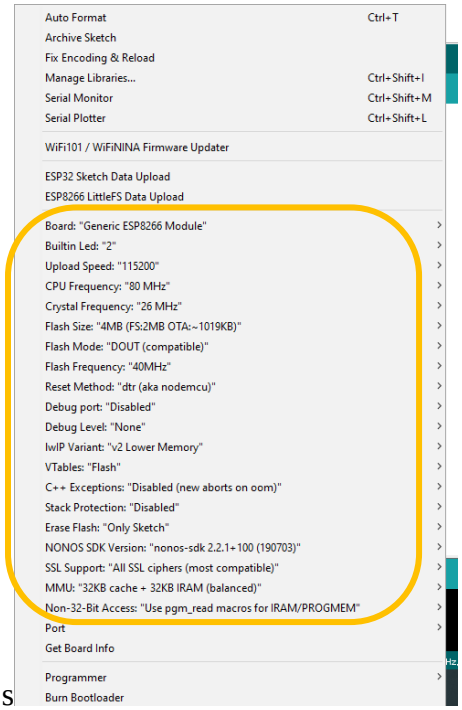
The core of the CanSat is the ESP8266. ESP8266 boards are able to read inputs from a sensor and turn them into an output. You can instruct your board on what to do by sending a set of instructions to the microcontroller on the board.

For the use of ESP8266 as On-Board Computer for the CanSat we need to install Arduino IDE along with the required Library Files.

IDE Setup

Setup of Arduino IDE for ESP8266/WEMOS (ESP12S)

1. Download the Arduino IDE from https://www.arduino.cc/en/software. Here for example procedure I have downloaded version 1.8.19.
2. For including the WEMOS/ESP8266 board, you can include the following in preference section. Go to File > Preference. The preference setting will open up and you will see **Additional Board Manager URL** option. Go there and paste https://arduino.esp8266.com/stable/package_esp8266com_index.json.
   This will download the board required for ESP8266.
   Ensure you will get following in tools section and make sure to have same configuration before uploading code.



3. Upload blink code which now you will get for ESP 8266 in the Examples.
   File >Examples > ESP8266 >Blink.

   If the buildin led blinks as per delay in code , Congratulations , you've successfully set the board for EDUCAN-MINI.

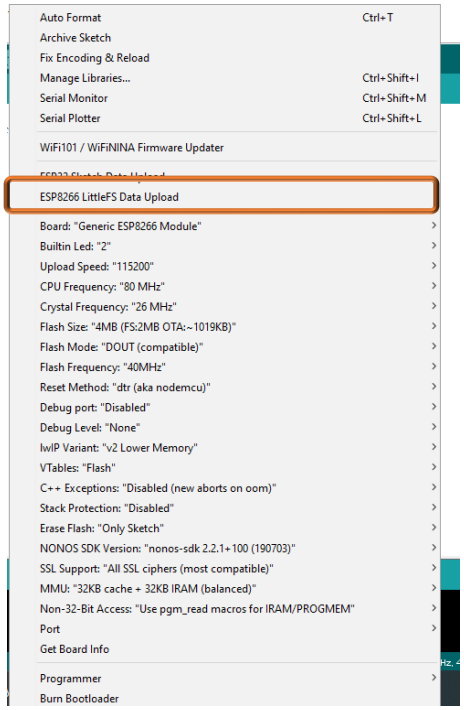
Setup for Little FS

1. Now for uploading the various file system that had been used for the extraction of sensor data , visualization of data in graphs , we have to use HTML , CSS, and Java Script. For this to integrate within the arduino as C++ format is nearly impossible and very difficult to debug. Thus the LittleFS is being used . For more understanding about the file system and Little FS along with SPIFFS , you can always visit to site https://arduino-

esp8266.readthedocs.io/en/latest/filesystem.html . For now , we will be conserning on the set up of Little FS for our example code .

1.1. Download the 2.6.0 or later version of the tool from https://github.com/earlephilhower/arduino-esp8266littlefs-plugin/releases
1.2. In your Arduino sketchbook directory, create tools directory if it doesn't exist yet.(you can Know your sketchbook directory by pressing CTRL+k).
1.3. Unpack the tool into tools directory (the path will look like <home_dir>/Arduino/tools/ESP8266FS/tool/esp8266fs.jar) If you are upgrading, overwrite the existing JAR file with the newer version.
1.4. Restart Arduino IDE.

2. Now here comes the most critical part , Here you should know that for uploading various file system in via Little FS , the procedure wants you to **place all file inside a directory named data.**
   2.1. Open the example directory which has been provided to you . the direcory is already in the format that Little FS require. (If you want to create a new code , modify the chart type or any other , format the file exactly same.) There you can see inside data directory, you have a basic HTML file , basic CSS file and a highchart.js source file for graph visualization.
   2.2. For uploading the directory called **data** , after you successfully install LittleFS , you will see following in your tools option.



2.3. To upload a LittleFS filesystem use Tools > ESP8266 LittleFS Data Upload. Make sure you have selected correct port , board and also any permission (if required).

Setup for arduino sketch file with the libraries and descriptions.

1. Now we can upload the .ino File that use the filesystem for executing the tasks .
   1.1. In case of this , you have to add additional libraries for communication of website to controller, use of WiFi card , for extraction of data from payloads .
   1.2. You can download library and add zip files via :
   Sketch > Include library > Add .zip library …
   Or include from arduino via :
   Sketch > Include library > Manage library… to open library manager.
   1.3. For Payloads MPU6050 , we have used **wire** library , for BMP , **Adafruit_BMP085** , **DallasTemperature** for DS18B20 library has to be installed via any of two installation processes in 3.2.
   1.4. ESPAsyncTCP and ESPAsyncWebServer has been used for communicating the ESP asynchronously with web server. Similarly **FS and LittleFS** library is used for use of LittleFS to open and execute the file previously uploaded as in procedure mentioned in **setup for littleFS.**
2. This is to clearify that any sensor can be deactivated or activated as per modification of code . by default the CANSAT have 3 sensors as payload along with the EPS providing the battery data. The sensors MPU6050 and BMP is configured to use I2C protocol while DS18B20 is following one wire protocol.
3. If any sensor data is not ment to show make sure to modify the function in .ino file along with HTML and CSS file too.
4. EPS is capable of accepting the additional power supply to charge the battery. By default it is using USB mini port for charging battery.

Setup for Host website

1. Connect the host to the AP of the EDUCAN-MINI. The OBC is used as the Access Point (AP). Thus no internet is required however the receiving device (host : where the data is to be obtained) is required to have connected to the AP of OBC .
   This can be modified in .ino file. By default ssid = "Dott" and password = "1234567890"
2. By default the softAP command set the IP address to 192.168.4.1
   However this can be set or changed. Read more via https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/soft-access-point-class.html

   The IP address is being published to serial monitor.

   PS: during image upload via LittleFS , serial monitor should not be opened.

Examples:

1. Reading battery

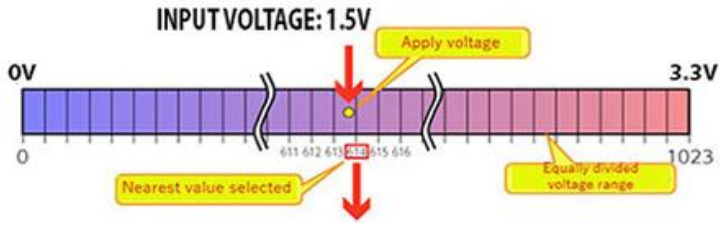
Code:

```
void setup() {

  // put your setup code here, to run once:
pinMode(A0, INPUT);
Serial.begin(9600);

}

void loop() {

int sensorValue=analogRead(A0);
float voltage= 2*sensorValue * (4.2 / 1023.0);
Serial.print("Voltage:");
Serial.println(voltage);
delay(1000);
}
```

Actual voltage is not an exact value such as 1V or 5V, but an irrational number, in which numerical values continue forever after the decimal point, such as 5.01342…V. These values cannot be directly handled by a computer. In electronic circuits, therefore, analog values are converted to digital values using a mechanism called AD converter so that the value can be handled by a computer. Arduino also has an AD converter by default, which can read analog voltages and convert them into digital values.
The AD converter divides the target voltage into several equal parts. For Arduino Pro Mini, voltages can be read in the range of 0-4.2V. These 0-4.2V values are divided into 1023 equal parts. The AD converter converts the analog voltage reading to the nearest equal value. If the voltage is 1.5V, for example, “614” is the closest value. This value can be sent to Arduino for use.

Formula:

$$\textbf{Voltage} = \frac{4.2}{1023} * \textbf{Sensor Value}$$



To ESP8266

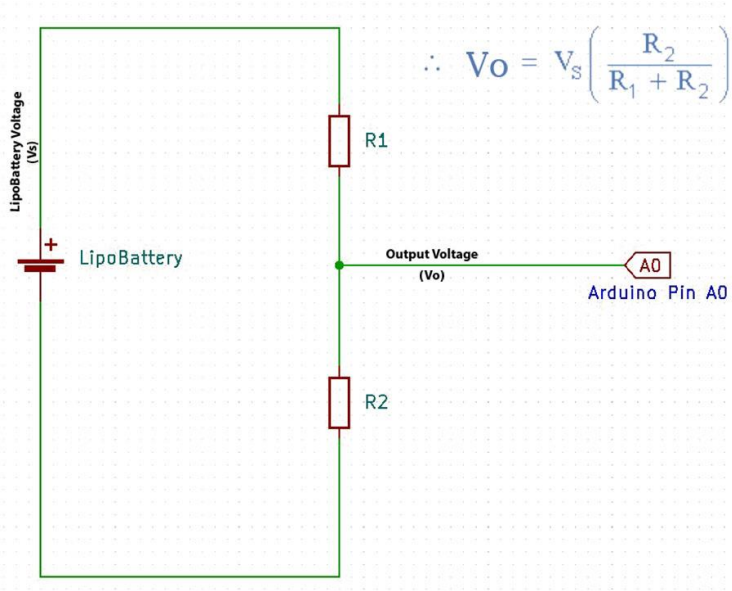


Fig: Voltage Divider Rule
To Calculate Voltage from A0 Pin

Output:

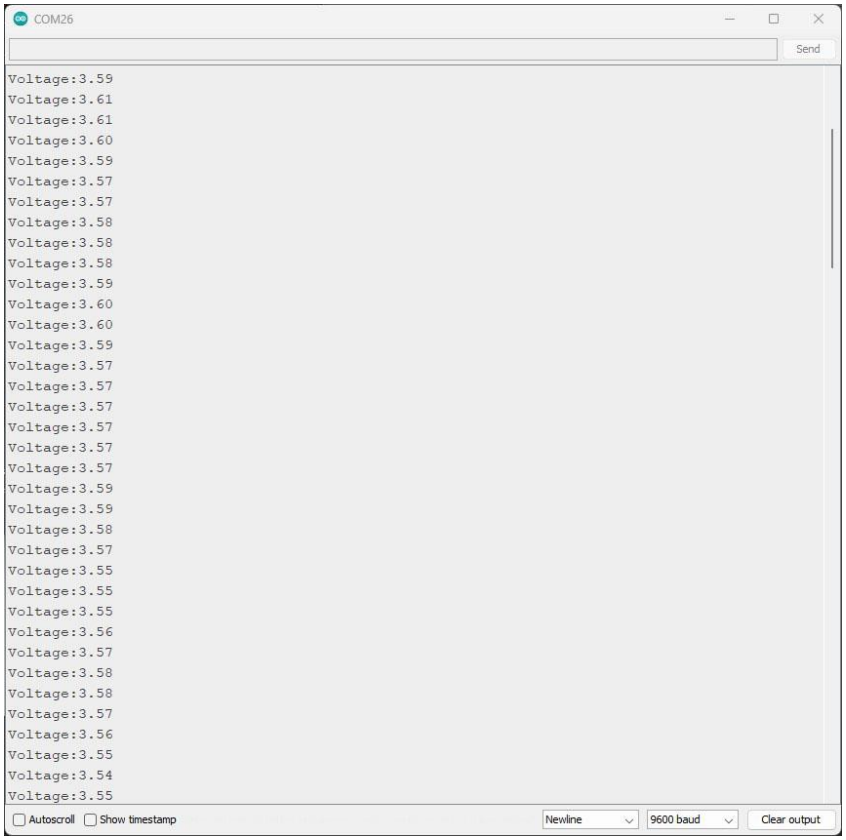


Fig: Battery Read Output

2. ESP8266 as Access point

Esp8266 has in-build Wi-Fi module which allows it to acts as both client and master. i.e it can also be connected to any prevailing Wi-Fi or can be configured itself to be a access point. Here is an example of using ESP8266 as an access point (AP).

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

#ifndef APSSID
#define APSSID "ESPap"
#define APPSK "thereisnospoon"
#endif

/* Set these to your desired credentials. */
const char *ssid = APSSID;
const char *password = APPSK;

ESP8266WebServer server(80);


void handleRoot() {
  server.send(200, "text/html", "<h1>You are connected</h1>");
}

void setup() {
  delay(1000);
  Serial.begin(115200);
  Serial.println();
  Serial.print("Configuring access point...");
  /* You can remove the password parameter if you want the AP to be open. */
  WiFi.softAP(ssid, password);

  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server.on("/", handleRoot);
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
}
```

3. Sending Payload data and Receiving (in WEB)

In this Example, we use the CanSat to gather Payload sensor data via various Sensor in our Payload Subsystem. Then, the gathered data are processed through the Onboard Computer (OBC) and then the data are transmitted using ESP8266.

Code for Read and Send Payload Data in Plain HTML format

```
#include <Wire.h>
#include <Adafruit_BMP085.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <ESP8266WebServer.h>

double startTime = millis();
double time_diff = 5000;
String upload_server;
WiFiClient client;

// Data wire is connected to the Arduino digital pin 4
#define ONE_WIRE_BUS 14

// Setup a oneWire instance to communicate with any OneWire devices
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature sensor
DallasTemperature sensors(&oneWire);

const char* ssid = "wemos";       // Set the SSID for the access point
const char* password = "asdfghjkl";  // Set the password for the access point

ESP8266WebServer server(80);

// BMP INSTANCES
Adafruit_BMP085 bmp;
// MPU6050 Slave Device Address
const uint8_t MPU6050SlaveAddress = 0x68;

// Select SDA and SCL pins for I2C communication
const uint8_t scl = D1;
const uint8_t sda = D2;

// Sensitivity scale factor respective to full scale setting provided in
datasheet
const uint16_t AccelScaleFactor = 16384;
const uint16_t GyroScaleFactor = 131;

// MPU6050 few configuration register addresses
```

```
const uint8_t MPU6050_REGISTER_SMPLRT_DIV   =  0x19;
const uint8_t MPU6050_REGISTER_USER_CTRL    =  0x6A;
const uint8_t MPU6050_REGISTER_PWR_MGMT_1   =  0x6B;
const uint8_t MPU6050_REGISTER_PWR_MGMT_2   =  0x6C;
const uint8_t MPU6050_REGISTER_CONFIG       =  0x1A;
const uint8_t MPU6050_REGISTER_GYRO_CONFIG  =  0x1B;
const uint8_t MPU6050_REGISTER_ACCEL_CONFIG =  0x1C;
const uint8_t MPU6050_REGISTER_FIFO_EN      =  0x23;
const uint8_t MPU6050_REGISTER_INT_ENABLE   =  0x38;
const uint8_t MPU6050_REGISTER_ACCEL_XOUT_H =  0x3B;
const uint8_t MPU6050_REGISTER_SIGNAL_PATH_RESET  = 0x68;

int16_t AccelX, AccelY, AccelZ, Temperature, GyroX, GyroY, GyroZ;

void setup() {
  Serial.begin(9600);
  Wire.begin(D2, D1);
  MPU6050_Init();
  bmp.begin();

  WiFi.softAP(ssid, password);  // Set up NodeMCU as an access point
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("Access Point IP: ");
  Serial.println(myIP);

  server.on("/", handleRoot);  // Handle root endpoint
  server.begin();
  sensors.begin();
}

void loop() {
  server.handleClient(); // Handle client requests if any

  sendToServer();
}

void handleRoot() {
  String html = "<html><body>";
  html += "<h1>CANSAT PAYLOAD</h1>";
  html += "<h2>Sensor Data:</h2>";

  String sensorData = ReadSensorData();
  html += sensorData;

  html += "</body></html>";

  server.send(200, "text/html", html);

}



String ReadSensorData() {
```

```
  // Replace with your actual sensor reading code
  double Ax, Ay, Az, T, Gx, Gy, Gz, vbat, sensor_temp, BmpTempr, BmpPress,
BmpAlti;
  String data;

  Read_RawValue(MPU6050SlaveAddress, MPU6050_REGISTER_ACCEL_XOUT_H);

  // Divide each with their sensitivity scale factor
  Ax = (double)AccelX / AccelScaleFactor;
  Ay = (double)AccelY / AccelScaleFactor;
  Az = (double)AccelZ / AccelScaleFactor;
  T = (double)Temperature / 340 + 36.53; // Temperature formula
  Gx = (double)GyroX / GyroScaleFactor;
  Gy = (double)GyroY / GyroScaleFactor;
  Gz = (double)GyroZ / GyroScaleFactor;
  BmpTempr = bmp.readTemperature();
  BmpPress = bmp.readPressure();
  BmpAlti = bmp.readAltitude();
  // Serial.print("Ax: "); Serial.print(Ax);
  // Serial.print(" Ay: "); Serial.print(Ay);
  // Serial.print(" Az: "); Serial.print(Az);
  // Serial.print(" T: "); Serial.print(T);
  // Serial.print(" Gx: "); Serial.print(Gx);
  // Serial.print(" Gy: "); Serial.print(Gy);
  // Serial.print(" Gz: "); Serial.print(Gz);

  sensors.requestTemperatures();
  Serial.print(" Sensor Temperature: ");
  sensor_temp = sensors.getTempCByIndex(0);
  Serial.println(sensor_temp);

  Serial.print("Battery Voltage: ");
  vbat = analogRead(A0);
  double mapped_vbat = map(analogRead(A0), 0, 1023, 0, 4.2);

  // String bmpData = read_BMP();
  // Example dummy values
  // double Ax = 1.23;
  // double Ay = 2.34;
  // double Az = 3.45;
  // double T = 25.6;
  // double Gx = 4.56;
  // double Gy = 5.67;
  // double Gz = 6.78;
  // double sensor_temp = 27.8;
  // double vbat = 3.7;

  String sensorData;
  sensorData += "<p>Ax: " + String(Ax) + "</p>";
  sensorData += "<p>Ay: " + String(Ay) + "</p>";
  sensorData += "<p>Az: " + String(Az) + "</p>";
  sensorData += "<p>T: " + String(T) + "</p>";
  sensorData += "<p>Gx: " + String(Gx) + "</p>";
```

```
  sensorData += "<p>Gy: " + String(Gy) + "</p>";
  sensorData += "<p>Gz: " + String(Gz) + "</p>";
  sensorData += "<p>Pressure: " + String(BmpPress) + "</p>";
  sensorData += "<p>Altitude: " + String(BmpAlti) + "</p>";
  sensorData += "<p>Sensor Temperature: " + String(sensor_temp) + "</p>";
  sensorData += "<p>Battery Voltage: " + String(mapped_vbat) + "</p>";

  return sensorData;
}

void sendToServer() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("Not connected to WiFi");
    return;
  }

  HTTPClient http;
  String url = "http://";
  url += WiFi.softAPIP().toString();
  url += "/";

  http.begin(client, url);  // Specify request destination
  http.addHeader("Content-Type",  "application/x-www-form-urlencoded");    //
Specify content type

  int httpCode = http.POST("");  // Send an empty POST request

  if (httpCode == HTTP_CODE_OK) {
    Serial.println("Data sent to server successfully");
  } else {
    Serial.print("HTTP Request failed with error code: ");
    Serial.println(httpCode);
  }

  http.end();  // Close connection
}

void I2C_Write(uint8_t deviceAddress, uint8_t regAddress, uint8_t data) {
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.write(data);
  Wire.endTransmission();
}

// Read all 14 registers
void Read_RawValue(uint8_t deviceAddress, uint8_t regAddress) {
  Wire.beginTransmission(deviceAddress);
  Wire.write(regAddress);
  Wire.endTransmission();
  Wire.requestFrom(deviceAddress, (uint8_t)14);
  AccelX = (((int16_t)Wire.read() << 8) | Wire.read());
  AccelY = (((int16_t)Wire.read() << 8) | Wire.read());
  AccelZ = (((int16_t)Wire.read() << 8) | Wire.read());
```

```
  Temperature = (((int16_t)Wire.read() << 8) | Wire.read());
  GyroX = (((int16_t)Wire.read() << 8) | Wire.read());
  GyroY = (((int16_t)Wire.read() << 8) | Wire.read());
  GyroZ = (((int16_t)Wire.read() << 8) | Wire.read());
}

// Configure MPU6050
void MPU6050_Init() {
  delay(150);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_SMPLRT_DIV, 0x07);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_1, 0x01);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_PWR_MGMT_2, 0x00);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_CONFIG, 0x00);
  I2C_Write(MPU6050SlaveAddress,  MPU6050_REGISTER_GYRO_CONFIG,  0x00);  //  Set
+/-250 degree/second full scale
  I2C_Write(MPU6050SlaveAddress,  MPU6050_REGISTER_ACCEL_CONFIG,  0x00);  //  Set
+/- 2g full scale
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_FIFO_EN, 0x00);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_INT_ENABLE, 0x01);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_SIGNAL_PATH_RESET, 0x00);
  I2C_Write(MPU6050SlaveAddress, MPU6050_REGISTER_USER_CTRL, 0x00);
}
```

The payload data are transmitted from Payloads. However The plain HTML isn’t capable of handling dynamic data . so the browser must be refreshed time and again for obtaining current data

Parachute for CanSat

Parachute is a device used to slow the motion of an object through an atmosphere by creating drag (air resistance). Parachutes are usually made out of light, strong, soft fabric originally silk, now most commonly nylon such that the components are not affected by final impact with ground after deploying.
There are different shapes of parachutes like Hexagonal, Semi-spherical, Cross parachutes, paragliding etc.

			
Parachute Types	Cross Parachute	Semi-spherical Parachute	Hexagonal Parachute
Drag Coefficient	0.60 - 0.85	0.42 - 1.5	0.75
Design Complexity	Simple	Complex and Time consuming	Very simple
Cost	Expensive	Very expensive	Relatively inexpensive

Size Calculation For Parachute

The size of the parachute is calculated using the following formula:

$$V_t = \sqrt{\frac{2m}{Cd.\rho.A}}$$

Where,
V_t = Descent velocity of the CanSat
m = mass of the payload
Cd = Drag coefficient of the parachute
ρ = Density of the air (around 1.255 kg/m^3)
A = Total area of the parachute

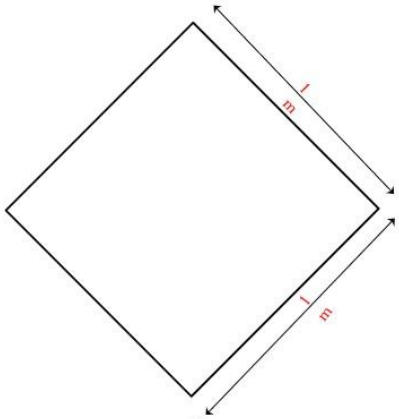
We deploy the CanSat from top of a building using a Parachute and record the data transmitted from the CanSat using our Ground Station Receiver.

Parachute Design For CanSat

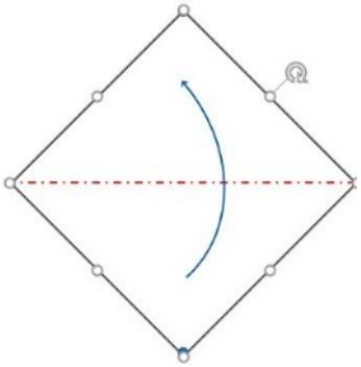
For the CanSat, we choose the Hexagonal Parachute because its design complexity is Simple and relatively inexpensive compared to other types of Parachutes.

Steps for Making Plastic Hexagonal Parachute

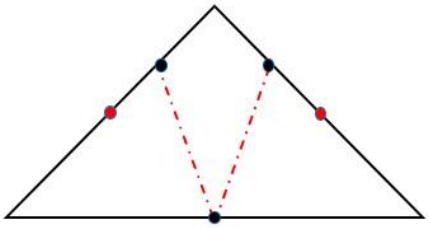
Step 1: Get a square shape plastic of dimension 1m × 1m.



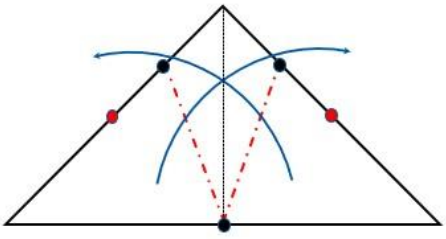
Step 2: Fold in half diagonal.



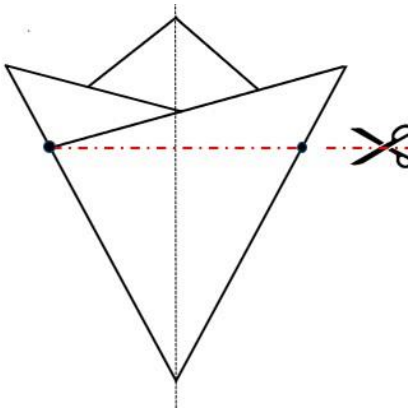
Step 3: Mark the points somewhere above the two upper midpoints (Near to the midpoint).



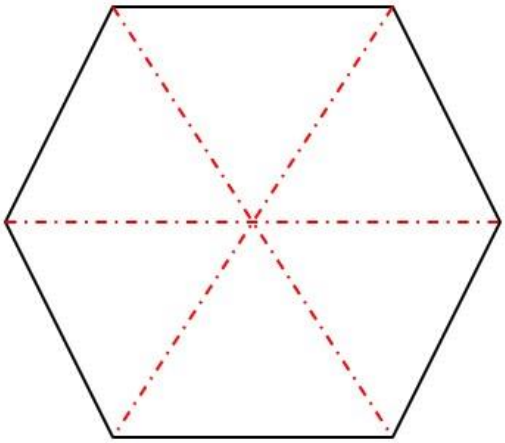
Step 4: Fold diagonally into thirds making the red dotted lines as an axis.



Step 5: Maintain the symmetry and cut from the red dotted line.

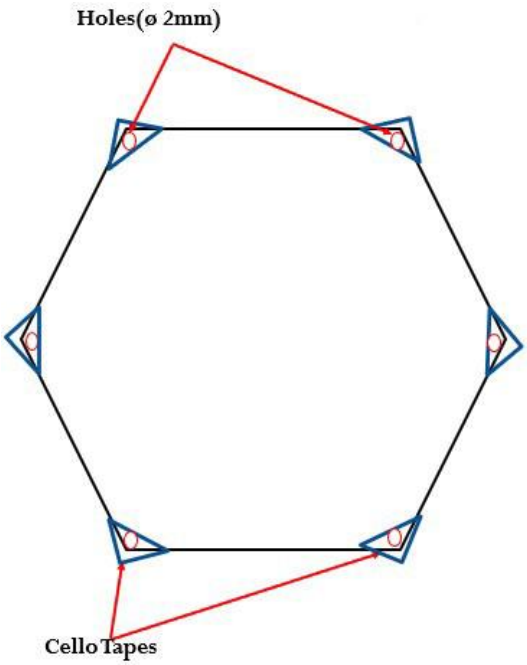


Step 6: Get larger piece of the plastic nd unfold it to get hexagonal geometry.



Step 7:

- Stick cello tapes on each of the vertices of hexagon.
- Make holes of diameter of 2mm in each vertices.
- Insert the strings to each of these holes and tie them.



Project:

Send Data through CanSat and Receive them in Ground Station (via web browsers)

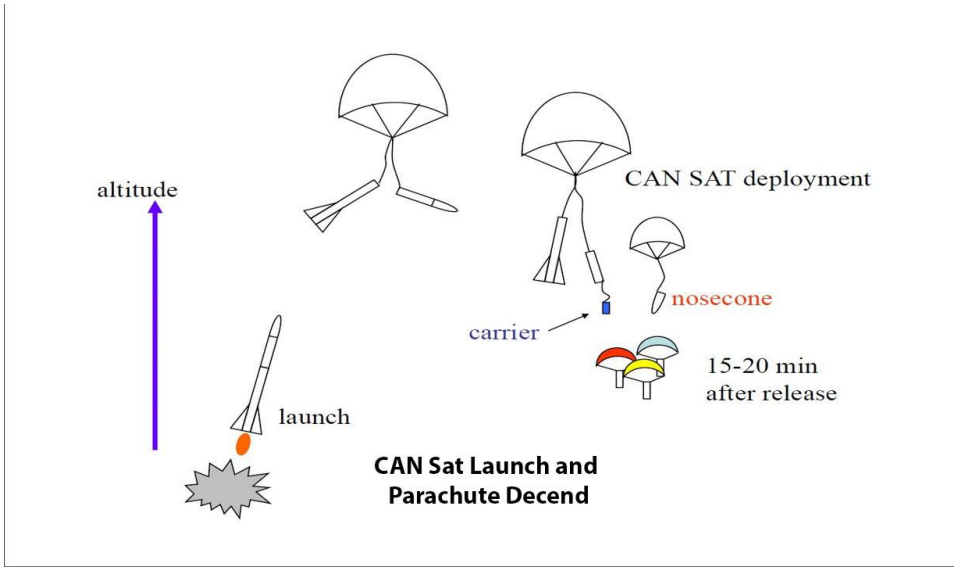
In this Project, we used the CanSat to gather various data through the various Sensors in the Payload Subsystem. Then, the various gathered data are processed through the Onboard Computer (OBC) and then the data are transmitted using Communication Subsystem which is embedded with OBC.

We have set a Ground Station, the ground station receiver (web servers) collect the data transmitted from CanSat. The received data can be stored locally or can be transmitted to the nearest device for analysis as well as displayed in the web browsers. For this , Make sure to connect the client device to IP where the ESP send the data .

Fig: CanSat

Send Data through CanSat and Receive them in Ground Station (via web browsers) using Parachute

Now for the final test, we take the CanSat or launch the CanSat to a certain height and drop it using a parachute. While the CanSat descends slowly with the help of parachute it continuously transmits with the help of ESP8266 in communication Board with various data from the sensors in the Payload Board. This data is then received by the Ground Station receiver.



For testing we took the CanSat to the top of the building and dropped it using the Parachute attached to the CanSat.