In-House Development of Instruments for Balloon Platform

Mayuresh Sarpotdar, A. G. Sreejith, Joice Mathew, S. Ambily, K. Nirmal, Ajin Prakash, Margarita Safonova and Jayant Murthy
Indian Institute of Astrophysics, Bangalore - 560034

Abstract

Some instruments that we put on a high-altitude balloon platform we develop in-house. They are used to either log the data from sensors, which are essential to be put on every launch, or to facilitate the separation of the payload from the balloon. The instruments we describe are:

1. Timer-based Flight Termination Unit
2. Geo-fencing Flight Termination Unit
3. Pressure and temperature data logger

In this document we describe the development of these instruments, give rough schematics and explain their operation and procedures for use.

Timer-based Flight Termination Unit

After a high-altitude balloon reaches its target height, it is expected to burst, and the payload is expected to come down in a controlled descent with the help of a parachute. In case the balloon does not burst, a failsafe technique is required to ensure the safe payload recovery with acquired data. A timer-based Flight Termination Unit (FTU) serves as this failsafe technique. It is an electronic circuit with a programmable timer and a Flight Termination Mechanism (FTM).

Hardware Implementation:

The layout is shown in Figure 1 and the FTM circuit diagram is shown in Figure 2. The FTU circuit is implemented on a printed circuit board (PCB). A rough layout for the PCB is as shown in Figure 3 and 4. A lithium polymer battery (7.6 V, 2100 mAh) is used to provide power to the circuit. The circuit and the battery are placed inside a pelican box which is attached to the load line under the parachute. The pelican box provides temperature insulation to prevent the FTU's temperature from dropping below minimum allowable limit. The programmable timer is based on an Atmega328P microcontroller unit (MCU) which triggers the FTM at a set flight termination time. The FTM circuit includes a relay which uses a 'thermal knife' to separate the payload from the balloon. The 'thermal knife' is a nichrome wire wound around the load line made of nylon. The nichrome wire heats up when current passes through it, melting the load line and thus separating the balloon from the payload. The nichrome wire is connected to the FTU by a normal wire which comes out from the pelican box (Figure 5) through the opening, sealed with M-Seal/Hot Glue for insulation.

Peripherals:

Different peripherals connected to the MCU are:

1. Removable LCD (Liquid Crystal Display)
2. 4 buttons
3. Relay circuit
4. Blinking LED (Light Emitting Diode)

The flight termination time can be programmed into the device just before the launch. The 16x2 (16 characters, 2 lines) LCD displays the current time and flight termination time. This LCD is removable so that after setting the flight termination time, it can be removed to reduce the weight and power load on the circuit.

The buttons are used to adjust the FTU settings. The flight termination time in hours, minutes and seconds, as well as the FTM active time in seconds (time for which FTM is powered up), can be adjusted by the buttons. The FTM active time is supposed to be long enough to ensure separation of balloon and payload (usually 30 sec).

Relay works as an electronically-controlled switch to connect the nichrome wire with the battery.

In absence of the LCD the blinking of LED ensures that the timer is on.

Software Implementation:

The following tools are used to program the MCU:

1. Arduino IDE (Integrated Development Environment) – a software where the C codes for the MCU are written and compiled to hex file.
2. An AVR-ISP programmer – an external hardware circuit which writes the hex file onto the MCU flash memory.

Operation:

The Arduino IDE has some predefined functions. We use the delay function to generate all timing sequences in the circuit. There are two modes of operation:

1. Counting mode.
2. Time-setting mode.

The default mode at start-up is the counting mode. In this mode the MCU updates the current time variables every 1 second acting as a clock, and copies them into the internal non-volatile memory EEPROM (Electronically Erasable and Programmable Read Only Memory). This is to keep a track of the current time and the flight termination time, so that if the MCU resets, it can still start counting from the last counted time.

In the time-setting mode the microcontroller allows the user to adjust the flight termination time and the FTM active time using the buttons.

LCD is refreshed in each mode. In time-setting mode it refreshes every 200 ms, and in counting mode it refreshes every 1 s. When the current time equals the pre-set flight termination time, stored in the EEPROM, the MCU activates the FTM which in turn separates the payload from the balloon.

Procedure before launch:

1. Before the launch, take the circuit out of the pelican box.
2. Connect the LCD to the circuit board.

3. Switch on the circuit and go to the time-setting mode.
4. Set values for the flight termination time (hours, minutes, seconds) and the FTM active time.
5. Shift back to the counting mode.
6. Remove the LCD and put the circuit back in the pelican case. Ensure that the LED is blinking, indicating the normal working of the FTU.
7. Proceed with the launch.

Geo-Fencing Flight Termination Unit

A high-altitude balloon has a tendency to drift away with lateral winds. It can drift up to 300 kms away from predicted burst location during the flight. In such cases, to avoid it from reaching undesirable locations (like sea, mountains, neighbouring states/countries), a failsafe mechanism is needed which keeps track of the current balloon location (in terms of latitude and longitude) and terminates the balloon flight if it goes outside of a pre-programmed 'geographical fence'. A geo-fencing flight termination unit adds this feature of geographical fencing to the balloon. It consists of a geo-fence, which works at high altitudes, and a flight termination mechanism (FTM).

Implementation:

The circuit for the geo-fencing FTU is implemented as an add-on 'shield' for an Arduino. An Arduino shield is a detachable circuit which can be connected on an Arduino microcontroller (MC) board header. We have developed this shield and its software for the Arduino Uno board. The geo-fencing feature is implemented using a Global Positioning System (GPS) module and the Arduino Uno board. The FTM consists of a relay which connects the nichrome wire with the battery. The Arduino Uno triggers the relay when the current location reported by the GPS is found to be out of the boundary defined by the programmed latitudes and longitudes. The relay passes the current through the nichrome wire, the nichrome wire heats up and cuts the loadline between the balloon and the payload. In such cases, the thermal knife is activated to separate the loadline between the balloon and the payload.

GPS module:

A GPS module from ublox (NEO-6) is used in this circuit. This module is selected because it works at high altitudes. This GPS module receives signals from different GPS satellites visible in the sky. Then it estimates its location with respect to the satellites and computes the latitude and longitude of the location. This process is called a GPS locking. Once locked, the GPS module updates its location every second and reports this location to a serial port in a specific format. This format is called NMEA (National Marines Electronics Association) sentence format. After decoding these NMEA format sentences, we can get the latitude and longitude of the location of the GPS receiver.

Microcontroller:

The Arduino MC decodes the NMEA sentences from the GPS module. The latitude and longitude of the launch location (Hosakote) is $13^0$04', $77^0$57'. We usually do not want the balloon to go further than 150 kms from the launch location which corresponds to around $1^0$15' shift in latitude and longitude. Therefore, the boundary is programmed as the square of geographical locations on the map described by $13^0$04' ± $1^0$15', $77^0$57' ± $1^0$15'. The current location of the balloon, obtained from decoded NMEA sentences, is then compared with the programmed boundary of latitude and longitude. If the balloon is outside the boundary, the MC triggers the FTM.

Removable LCD:

The removable LCD displays the current latitude and longitude. It is useful to see whether the GPS is locked to a proper location while on the ground and whether the decoding process is working fine. The LCD is removed before the launch to reduce the weight of the circuit and the power load on the circuit.

Flight Termination Mechanism:

The FTM used in this circuit is similar to the FTM used in the timer-based flight termination unit.

Procedure before launch:

1. Connect the Arduino with the computer and reprogram with flight specific data such as: FTM active time and latitude/longitude of the boundary for geo-fencing.
2. Power on the circuit and check whether the GPS in the circuit is locked on the current location.
3. Put the circuit inside the payload box and place the switch for the circuit outside the box.
4. Switch on the circuit just before launch.

Pressure and temperature data logger

A generic set of housekeeping sensor data needs to be recorded in every high-altitude balloon flight. A pressure sensor log is a marker of the altitude of the balloon as an alternative to the GPS logs. A temperature sensor log is useful to get an idea of instruments operating temperature during the flight.

Hardware Implementation:

The circuit consists of microcontroller unit (MCU), sensors, storage memory card, graphical LCD and power circuit. These components are implemented on a PCB. See Figure 6 for the layout of the PCB, and Figure 7 for the working data logger circuit. The PCB and the battery are stored in a 3D-printed box (Figure 8, Figure 9 and 10), which has openings to attach the LCD and a mini-USB cable for charging and for the ON/OFF switch.

Microcontroller:

The microcontroller used in this circuit is an Atmega328P (SMD version). It is responsible for interfacing all other components, including sensors, memory card, and LCD. It also generates timing sequences to record the sensor data at regular intervals.

Sensors:
The circuit consists of a pressure sensor (BMP180) and a temperature sensor (LM75). Both sensors have a data communication interface called I2C (Inter Integrated Circuit). The MCU connects to these sensors on I2C interface and reads the data. The data from the temperature sensor is in decimal format with a resolution of 0.5^0C. Hence it can be stored on the SD card directly. The pressure data is in raw format in terms of ADUs and needs to be further converted into pressure value in millibars.

Storage memory card:
A micro-SD memory card is used to store the pressure and temperature data collected during the flight. This is removable, and after the flight the data files for pressure and temperature can be copied directly into the computer. A 2 GB micro-SD card is sufficient to store 48 hours of data.

Graphical LCD:
The LCD is used to see the sensor values on-the-go. This LCD is removable and is removed before the launch. It is very useful for debugging purposes.

Power:
The circuit is powered by a single cell lithium polymer battery (3.7 V, 2100 mAh). The circuit draws ~50 mA current with the LCD, and ~40 mA current without the LCD. Hence, the battery is assumed to last for ~48 hours, which is sufficient for a typical balloon flight. The charging circuit is on-board and the battery can be recharged with a mini-USB cable.

3D-printed housing:
A 3D-printed box was designed (Figure 10) to keep the circuit and the battery together for the flight. This box is very lightweight (~20 gms) as compared to an aluminium box or any other box of the same size. In addition, since the data logger is kept inside the payload box during the flight, where other electronics components generate enough heat, the temperature insulation is not needed. Moreover, we get a log of the ambient temperature of other electronic circuits in the payload box.

Software Implementation:
The MCU generates a 5-second time pulse. After every 5 seconds, it queries both temperature and pressure sensors for the new data. After obtaining the data, the MCU converts it to the text format and saves it on the SD card. The SD card is formatted in a FAT32 file system, and the MC saves the data in .txt files meant for the same file system. This allows easy access to the stored data after the launch. We can remove the SD card, put it in an SD card reader connected to a computer, and plot the data directly without any further processing. An

approximate value of the altitude is obtained from the pressure value, measured by the sensor, using the following barometric equation:

$$altitude = 44330 * \left(1 - \left(\frac{p}{p0}\right)^{\frac{1}{5.255}}\right),$$

where p is the measured pressure and p0 is the pressure at sea/reference level.

How to use:

The data logger can be used as is in flight, without any reprogramming. A sampling time of 5 sec is ideal in terms of the battery life, as well as the file sizes, to enable logging of 24 hrs temperature, pressure and altitude data. The altitude data can be used as a cross-reference for altitude values obtained from the GPS data log. The circuit along with the battery weighs less than 50 gms, and can be easily added to any payload configuration.

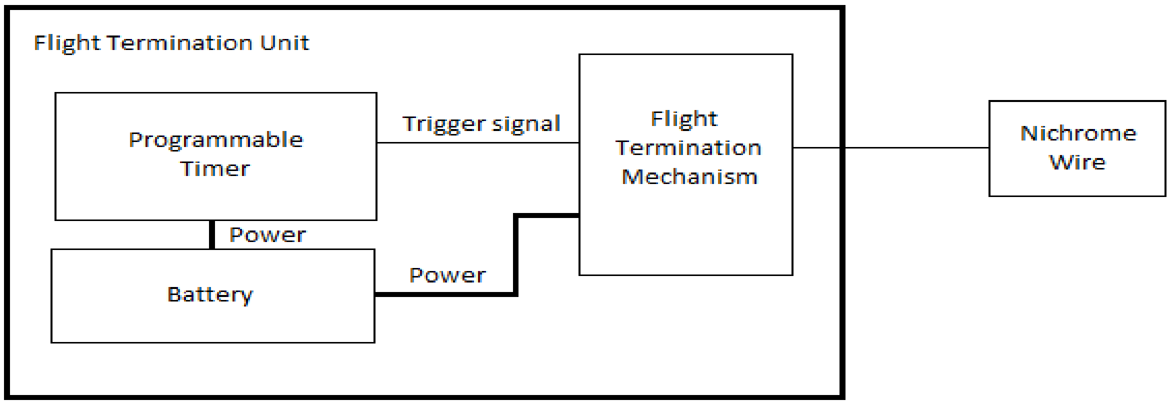


Figure 1. Diagram of the Flight Termination Unit

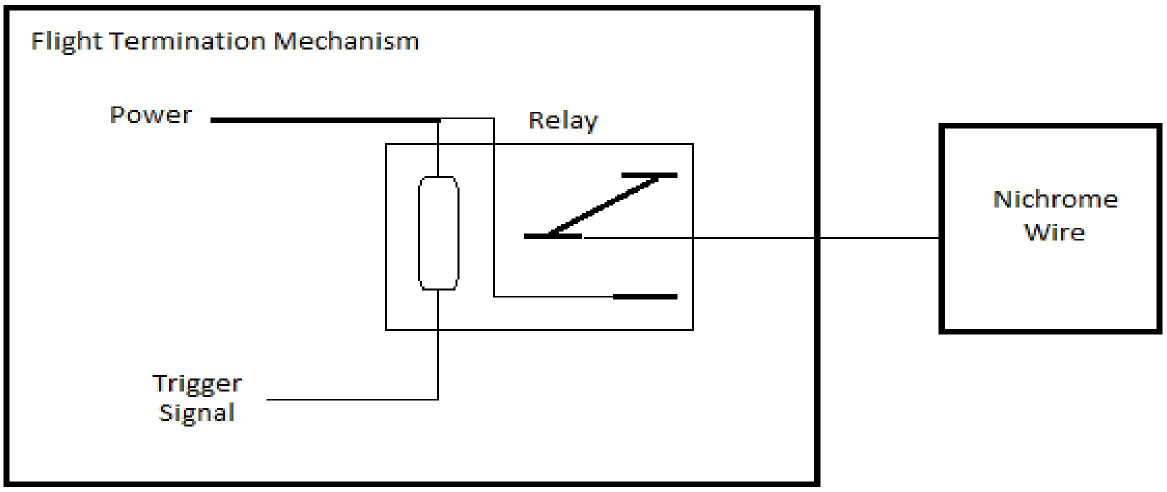


Figure 2. Circuit diagram of Flight Termination Mechanism

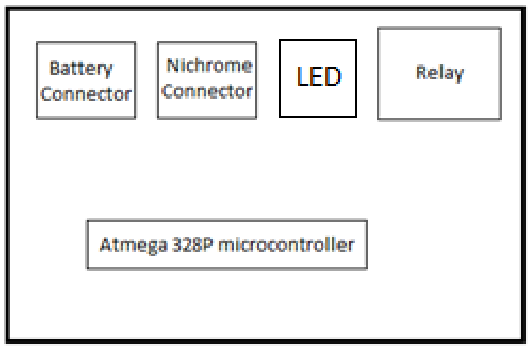


Figure 3. Front side of the PCB

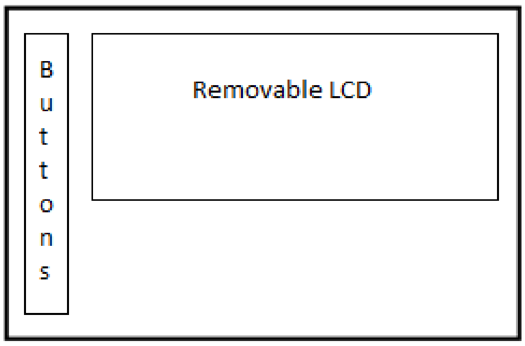


Figure 4. Back side of the PCB

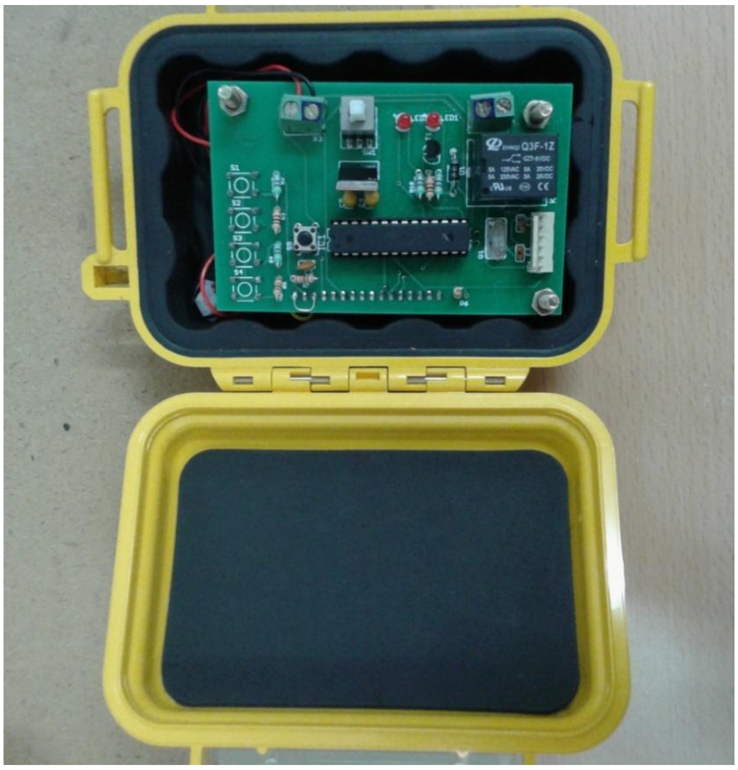


Figure 5. FTU inside the pelican case

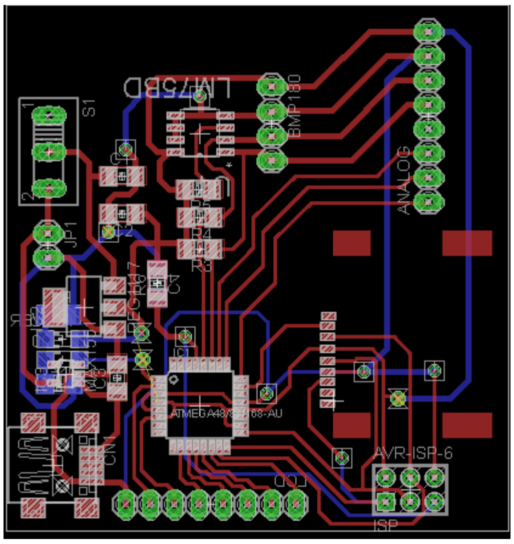
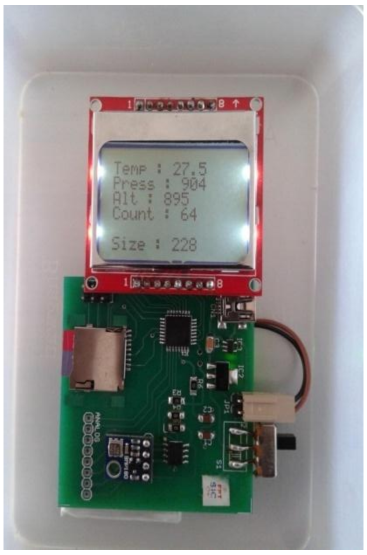


Figure 6. Layout of data logger PCB



```
Temp : 27.5
Press : 904
Alt : 895
Count : 64

Size : 228
```

Figure 7. Working circuit with LCD, SD card and battery connected

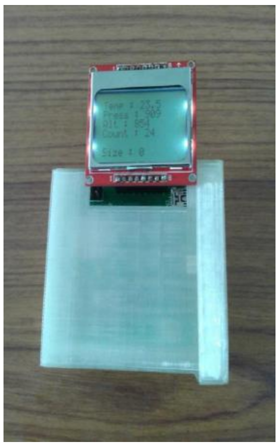


Figure 8 Circuit inside 3D printed box with attached LCD

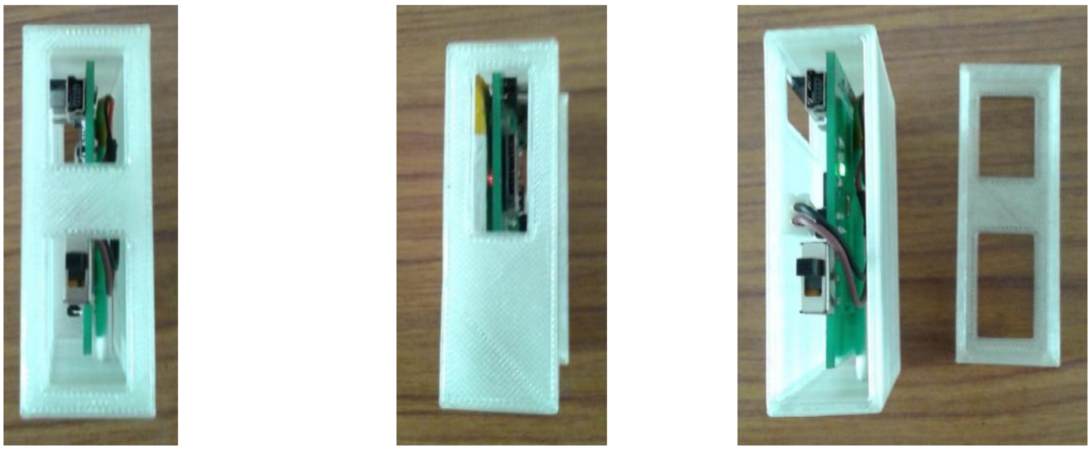


Figure 9 Different views of the 3D printed box

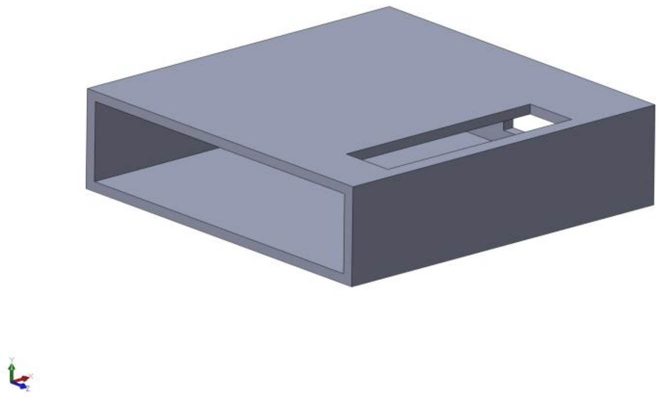


Figure 10 CAD model of 3D printed box