

Logic of program explain according to the order of the code

Define

The pins are defined following the hardware design.

Setup phase

* Pin mode setting for outputs and inputs
* Initialisation of the output pins; all are normally set to 0.
* Buzzer indicating that the system has been powered on. The indication signal is a buzz of 0.25s
* I2C initialization of the EEPROM. Most EEPROMs run at 400kHz.
* Check the status of the battery, the sensor, the e-match of the drogue and the e-match of the main.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wrong status | Power | Sensor | ERPORM | Ematch |
| Buzzer indication | Dot dot dot | Dah dot dot | Dah dah dot | Dah dah dah |

1. Considering the purpose of some tests, there are times when it is not necessary to check the battery voltage (when powered by the laptop tested through laptop)

#define battery\_threshold  999                                //

The battery is rated at 8.4V and we define that the PSI will indicate the wrong status if the battery voltage is below 8.2V. barrery\_threshold is defined as 999,( 8.2/8.4\*1024=999). Once the voltage measured using analogRead() is less than barrery\_threshold, it means the battery voltage is less than 8.2V.

1. Considering the purpose of the test, there are times when Ematches are not connected and should not prevent other code from running, e.g. PSI is tested without Ematches in the drone test.

* Set the EEPROM settings as follows:
* myMem.setMemorySize(512000/8);                                        //In bytes. 512kbit = 64kbyte
* myMem.setPageSize(128);                                              //In bytes. Has 128 byte page size.
* myMem.enablePollForWriteComplete();                                  //Supports I2C polling of write completion
* myMem.setPageWriteTime(3);                                           //3 ms max write time

1. Memory size, for EEPORM 24LC256 has 512K bits = 512k/8 bytes =64k bytes.
2. Page size.
3. Supports I2C polling of write completion
4. Sets the maximum write time.

* Initialization of key variables and parameters

1. Read initial air pressure which is the base pressure for calculating the reference pressure
2. Set Mode to Ready to Launch Mode .
3. The ready to launch Mode beep timer starts. The beep in this mode is an indication of this Mode. The beep appears every three seconds.
4. Read EEPORM maximum memory capacity
5. Set two identical sample\_size

Operation cycle

* Starting from mode 1.
  1. The sensor continuously reads the altitude and PSI constantly checks the altitude. The altitude and relative time are only recorded into the EEPROM if the current altitude is above the launch\_threshold.
  2. Once the EEPORM has written one data (altitude or time), the address for EEPORM should be incremented by byte size, 4, using EEPROM\_Check() from PSI.h. Everytime the address of EEPROM is updated, EEPROM will be check if it is fully loaded.
  3. Indication of this mode.
  4. When the launch is detected, the operation will go to the next mode.
* How to measure time.

The method for obtaining the time data is obtained using the millis() function. millis() measures the absolute time, starting when the Arduino is first powered on. To obtain the relative time to be recorded in the EEPORM, the reference time is set when power-up is detected. millis() function returns a value in milliseconds, which is converted to a value in seconds and stored in a variable called time\_s\_float.

* Move to the mode 2, where powered fight happens and the apogee detection happens.

1. There are variables needs to be initiated amd reset in this loop.
2. read\_altitude is the array that contains the altitude that reads from the sample.
3. read\_time is the array that contains the time that reads from the sample.
4. A struc named read\_time\_altitude contains the data set of read\_time and read\_altitude
5. An array called buffer\_sum\_gradient contains 2 elements to store the gradients of 2 samples in 1 buffer. They are reset to gradient\_init in each loop.
6. The the sum\_gradient is reset to 0 for each loop.
7. After resetting, a dataset of time and altitude is stored in the EEPORM. In such a dataset storage an updated time\_s\_float and an updated relative\_altitude are obtained. The current time is read from time\_s\_float and the current altitude is read from relative\_altitude. They are put into the struct time\_altitude and the time\_ altitude is put into the window in which apogee detection occurs. This window contains the data set whose has the number of dataset equal to two sample\_size .(A window has 30 dataset for sample\_size of 15). The window acts as a buffer, constantly reading the dataset and using a first-in-first-out (FIFO) implementation. the FIFO ensures that the window has a constant number of datasets and is always updated with new ones.
8. When the window is full, the apogee detection starts to work.
9. We used two gradient as the determination to decide the apogee. In the first sample, the time and the altitude are separately stored in two arrays, read\_altitude and array\_time so that the apogee detection algorithm can seperatedly read them.
10. The apogee detection algorithm used the method called linear regression method. It needs time and altitude to calculate its average gradient with the least impact from noise.
11. To get the specifical data at a certain order, the peekIdx(&sample, index) function is used. peekIdx(&sample, index) returns the value at index th order of the sample without making any changes to the sample.
12. The calculated gradient of first sample returns to buffer\_sum\_gradient[1] and that of the second sample returns to buffer\_sum\_gradient[0].
13. The pop(&somewhere) function is used to pop the first/oldest data in the window. A data set is created to store the popped data.
14. Only if both value of buffer\_sum\_gradient[1] and buffer\_sum\_gradient[0] are below the gradient\_threshold, we can decide the apogee is detected. We use 2 determinations to decide the apogee to minimize the impact of noise that is off the track.
15. A higher than usual height of noise may result in the sample containing the noise having an incorrectly calculated gradient, but the next sample without such noise will have the correct calculated gradient, which will correct the decision based on the noise.
16. A special altitude marker is stored to mark the deployment of the parachute. This special altitude marker is 1000 times the original altitude so that this altitude can be recovered in the post-flight data processing.
17. Considering the eamtch needs the time of iginition\_duration to be ignited, at this time the data storing continues to happen so that there is no gap at this time.

* Move to mode 3 where only the alaitude of main parachute keeps being detected.
  1. In mode 3, normal data storage continues in this loop until the current altitude is detected as being below main\_threshold.
  2. Once main\_shreshold is detected, a special marker for the main deployment altitude is stored as 1000 times the original altitude and the main parachute match is ignited.
* Move to mode 4 where the land conditions are checked.

1. Normal data storage occurs continuously in this loop and in the EEPORM\_Check() function, if the updated address exceeds EEPORM\_length, the address will always be set to EEPORM\_length - 4. The maximum address that can be reached is always EEPORM\_length - 4. Any data after that address will all go to that address without affecting other data.
2. The address overload condition is if the update is less than EEPORM\_length - 5. If so, it means that the address has not yet reached the maximum length of the EEPORM. This condition will determine if EEPORM should stop loading.
3. Another condition for landing is if the current altitude is low land\_threshold. This condition will determine if the rocket has landed.

* Move to mode 5 where nothing works but buzzer indicating this land status either the EEPORM has fulled or the the land condition has met.
  1. Nothing will be running except the buzzer indicating this mode.

Adjustable constant parameter values explained

* Ignition\_duration is the period for the drogue ematch and main ematch to be ignited.
* main\_release\_a is the altitude where the main parachute shou be deployed. In meters.
* The implementation for the queue is set to FIFO (First In First Out).
* Launch\_threshold is the threshold altitude to decide the launch.
* Land\_threshold is the threshold altitude to decide the rocket has landed.
* The reading\_rate is the rating at which the altitude data is read from the sensor and written to the EEPORM. During the powered flight phase the read rate is 60, determining that there is an approximate frequency of 20 Hz. The read rate is doubled during the descent phase, making data storing slower as the rocket descends at a lower speed than when rocket ascends.

#define reading\_rate  60

* A window of approximately 30 data should be available for 1.5 seconds. The reading rate of the sensor/EEPORM is approximately 20 Hz. A window of 1.5 seconds means for PSI that the apogee is detected 1.5 seconds later than the actual apogee. This is because the window contains the most recent data. The moment the apogee is detected in the current window means that the rocket has been descending from the actual apogee for the time that the window contains.

#define sample\_size  15

* To minimise the effect of detected delays at apogees. In the normal condition, the gradient of an sample is actually the vertical velocity of the rocket. When the vertical velocity drops below 0, it can be assumed that the rocket is descending after apogee. However, if the apogee threshold is slightly greater than 0 and the vertical velocity reaches that positive apogee threshold near 0, the rocket can be considered almost apogee. Considering that apogee is always detected late, it can be concluded that the rocket has actually reached apogee orat least near the apogee.
* Based on our previously documented ground tests, we can say that an apogee\_threshold of 1.5 is an appropriate value.
* And the value of gradient\_init should always be greater than apogee\_threshold, as it is the reset value of gradient in the mode 2 loop, where apogee detection is repeatedly run.

#define apogee\_threshold 1.5f

#define gradient\_init 2.0f

External Library reference

<SparkFun\_External\_EEPROM.h> <https://github.com/sparkfun/SparkFun_External_EEPROM_Arduino_Library.git>

<cppQueue.h>

<https://github.com/SMFSW/Queue.git>

"MS5611.h"

<https://www.arduino.cc/reference/en/libraries/ms5611/>