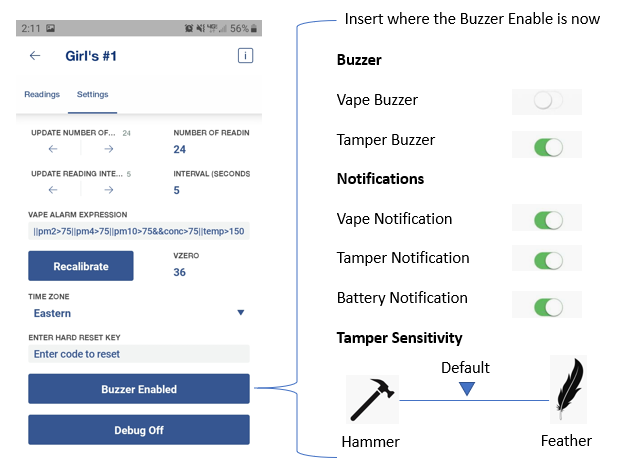
**Software function specification for Zeptive firmware, modifications built off of version 112a new version will be 114.**

1. **The following are the tasks needed to use a running average to filter particle counts to reduce noise on the**
   1. TBD
2. **The following are the tasks needed to use a running average to filter battery signal to reduce noise on the signal** 
   1. TBD

1. **The following are the tasks needed to reduce delay for the accelerometer audible from 5s to 1s or 2s**
   1. TBD
2. **The following are the tasks needed to switch to voltage from % (?) May be solved with fuel gauge**
   1. TBD
3. **The following are the tasks needed to make battery alerts for a duration time or number of connection events**
   1. TBD
4. **The following are the tasks needed to use the RTC to turn off overnight to improve battery life**
   1. TBD
5. **The following are the tasks needed in the firmware to implement Particle PoE as the power module for the system**
   1. TBD
6. **The following are the tasks needed to implement the system on-time in FRAM to check on/off time of sensor**
   1. Set a global variable CycleOnTime and set to zero on startup
   2. Create a new FRAM variable in the state structure OnTime
   3. Reset OnTime to Zero every time we reset FRAM
   4. On setup grab the current timestamp save to CycleOnTime
   5. On entering Sleep function for either (standby or deepsleep) get the elapse time for this session. Add to the OnTime in FRAM and save back in FRAM
   6. Save FRAM state before entering sleep
7. **The following are the tasks needed to get the date and time from Blynk**
   1. TBD
8. **The following are the tasks needed to implement vape alert duration**
   1. When a vape alert occurs I set a VapeAlertTime to the current time
   2. At the end of the vapor alert, I calculate the elapse time.
   3. Since no Vape end message is sent I just log in debug the Vape Alert duration
9. **The following are the tasks needed to implement the changes into the firmware to use the 6 new pins developed by Blynk.**



* 1. Define new variables

bool buzzerTamper

bool buzzerVapor

bool notifyVapor

bool notifyBattery

Int settingTamper

**Rename variable**

buzzerActivated to buzzerTamper

* 1. **Create new pins**

buzzerVapor Boolean on Blynk side = V27

notifyVapor Boolean on Blynk side = V28

notifyBattery Boolean on Blynk side = V29

settingTamper Boolean on Blynk side = V32

settingTamper integer on Blynk side = V31

**Rename pin**

buzzerActivated to buzzerTamper Boolean on Blynk side = V24

* 1. **In InitalizeAppWidgets() Function add the following:**

Write the value of buzzerTamper to pin V24

Write the value of buzzerVapor to pin V27

Write the value of notifyVapor to pin V28

Write the value of notifyTamper to pin V29

Write the value of notifyBattery to pin V32

Write the value of settingTamper to pin V31

* 1. **In BLYNK\_WRITE(V22) Function add the following:**

**This function is called when the Reset Expression is called from the server with the correct key “resetme”**

Reset to default and then write the value of buzzerTamper to pin V24

Reset to default and then write the value of buzzerVapor to pin V27

Reset to default and then write the value of notifyVapor to pin V28

Reset to default and then write the value of notifyTamper to pin V29

Reset to default and then write the value of notifyBattery to pin V32

Reset to default and then write the value of settingTamper to pin V31

**Default**

    state.ActivityThreshold=100; // Which is set to 1 in Blynk

    state.buzzerTamper=true;  // Buzzer Tamper

    state.buzzerVapor=false; // Vapor Buzzer

    state.notifyVapor=true; // Vapor Notify

    state.notifyTamper=true; // Tamper Notify

    state.notifyBattery=true; // Battery Notify

* 1. **Create a new BLYNK\_WRITE() function for pin V27**

This will set the state of the local variable buzzerVapor to be what is passed from Blynk in pin V27 and then save in FRAM

In Debug mode will beep twice if this function is called

* 1. **Create a new BLYNK\_WRITE() function for pin V28**

This will set the state of the local variable notifyVapor to be what is passed from Blynk in pin V28 and then save in FRAM

In Debug mode will beep twice if this function is called

* 1. **Create a new BLYNK\_WRITE() function for pin V29**

This will set the state of the local variable notifyTamper to be what is passed from Blynk in pin V29 and then save in FRAM.

In Debug mode will beep twice if this function is called

* 1. **Create a new BLYNK\_WRITE() function for pin V32**

This will set the state of the local variable notifyBattery to be what is passed from Blynk in pin V30 and then save in FRAM.

In Debug mode will beep twice if this function is called

* 1. **Create a new BLYNK\_WRITE() function for pin V31**

This will set the state of the local variable settingTamper to be what is passed from Blynk in pin V31 and then save in FRAM.

In Debug mode will beep twice if this function is called

When this button is called reinitialize the accelerometer and configure using a passed variable instead of a hard coded value.

If settingTamper is set to 0 then set accelerometer value to 144

If settingTamper is set to 1 then set accelerometer value to 100

If settingTamper is set to 2 then set accelerometer value to 72

***Best way to reset accelerometer is to reset the system so will be done at this point***

* 1. **Change BLYNK\_WRITE(24) function**

This will set the state of the local variable buzzerTamper to be what is passed from Blynk in pin V24 and then save in FRAM

* 1. **Modify the main code so that**

Everywhere buzzerActivated is used change to buzzerTamper

Modify the logic that turns the buzzer on after a vape alert to first check the buzzerVapor flag and only turn the buzzer on if it is true. Do not touch the turn buzzer off flag it will not hurt if it is off and the application shuts it off again. Want to eliminate a race condition where the flag changes during a buzzer action and it never shuts off

Modify the logic that sends a notification after a vapor alert to first check the notifyVapor flag and only send the notification if it is true. Also modify the logic that sends a notification after a vapor alert end to first check the notifyVapor flag and only send the notification if it is true

Modify the logic that sends a notification after a tamper alert to first check the notifyTamper flag and only send the notification if it is true. Leave all other logic in place we already removed the notification ended message.

Modify the logic that sends a notification after a battery alert to first check the notifyBattery flag and only send the notification if it is true. Leave all other logic in place we already removed the notification ended message.

Modify the logic that configures the accelerometer and pass a variable instead of a hard coded value.

Values will be as follows:

1 = Tap

2 = Default

3 = Bang

If settingTamper is set to 1 then set accelerometer value to 72

If settingTamper is set to 2 then set accelerometer value to 100

If settingTamper is set to 3 then set accelerometer value to 144

Read this flag before initializing the accelerometer and use new value.

**NOTE: Debug can’t be on when changing settings page!!**

**Support for Over the Air (OTA) firmware updates and change to architecture to connect only on alerts for Version 1 Hardware**

* Based on version 115 Zeptive firmware and includes:
* Enable the Real Time Clock (RTC) so devices can connect at a configurable time frame to check for firmware updates and connect to server at a configurable time frame to check system health
* Enabling the Watchdog Timer to turn the unit and connect to servers at a preconfigured time.
* Only connect to Particle and Blynk if a tamper or vape alert occurs
  + We will need to connect prior to the actual alert in order to make sure there is no lag in connectivity. To do this we will need a new warning threshold at 50% of alert threshold for tamper and vape to trigger connection to Particle and Blynk.
    - For an example of tamper, if tamper is set to 100 then at factor 50 we would need to connect to the servers
    - For an example of vape, if vape settings are pm 1 at 300, pm 2 at 300, pm 4 at 300, pm 10 at 300 &&conc at 100 then at 150/150/150/150&&50 we would connect to the servers
      * Concern here is that our alarms only last for a few seconds <30sec which is a shorter time than the connection times for most of our devices
    - A way to locally buffer data in case we cannot connect in a timely fashion then send the data when we do connect
* Provide a test mode where the device does connect to the servers and transmit data
  + Tradeoff discussions:
    - Could be connected to the Debug button or a new button? We probably want it connected to Debug because otherwise we will not be transmitted via debug in this new architecture with alert only data transmission.
    - Could turn the unit on all the time and connect to servers all the time? Probably, not desirable as this will kill the battery if someone forgets to turn debug off
    - Could turn the unit on and connect to server then sleep after 2 min as we do now? Probably the easiest to implement but does require us to trigger the PIR to turn the unit back on. This would mitigate the risk though of someone turning debug on and forgetting to turn it off.
    - Is there ever a time we want the unit to stay on indefinitely? OTA firmware should keep the unit on because it can override application firmware.
* Average the battery signal to reduce noise
  + Require multiple drops below threshold to generate battery alert sent to Blynk

**Support for Version 2 Hardware**

Hardware changes:

* Changed Spec board design to move electronics off the adapter board and put it on the Spec daughter board in order to reduce noise on the signal. The Spec daughter board was expanded to allow for other types of Spec sensors.
* The ADXL accelerometer was moved to the adapter board to simplify the design and reduce costs
* A fuel gauge was added for the system reserve battery (4400mAh battery) to replace the rough calculation in the firmware. The calculation results in some error in the measurement and the WiFi module does not have a fuel gauge as it does in cell.
* A fuel gauge was added for the system battery (15000mAh battery) so users can tell when the system battery is getting low.

Firmware changes

* Test new Spec module to confirm performance of sensor design
  + May need to average the Spec signal to further reduce noise on the channel.
* Test integrated accelerometer to confirm performance of sensor design
  + May need to adjust values for tap/default/bang based on integrated accelerometer
* Add support in firmware to trigger fleet wide OTA updates as recommended by Particle. Requires some devices to be set up as Products in Particle and labeled as development devices.
* Firmware to get battery charge %
* Implementation of fuel gauges will require a weighting algorithm to display battery percentage to users on dashboard.
  + 90 (4400mAh)/10 (15000mAh) weighting until reserve battery (4400mAh) is completely depleted then only show system battery charge %
* **We want to learn how to implement these fuel gauges BUT we may jump right to deploying Version 3 hardware where the battery architecture is completely changed!! Need to evaluate best use of time for this implementation.**

**Version 3 Hardware Changes (Specifications TBD – no need to work on these items now):**

Overview of Changes

|  |
| --- |
| **Version 3a Solo Sensor** |
| Remain with Particle Wi-Fi |
| **Custom PoE** |
| Remain with Particle Cell |
| **Move to custom PC board** |
| Particle OTA |
| **Same:** PIR, Accelerometer, Audible, SPEC, Sensirion |
| Blynk |
| Custom Case |
| Always ON Battery |

Electronics Specific Changes

|  |
| --- |
| **Version 3a Solo Sensor** |
| Direct power via USB |
| Watchdog Timer |
| Real-Time Clock |
| Keep 1 extra ethernet |
| Keep small reserve battery for real time clock and SPEC |
| Change battery configuration to start with larger battery; monitor charge here |
| Keep on/off |
| System LED for battery/connection/hardware status, troubleshooting, typically off if no issues |
| Includes design changes from Version 2 |
| Keep MicroSD – near battery door |
| I/O expander board control and monitor? |
| Buzzer could be installed inside the case or board mounted - TBD |
| Remove Solar |
| Remove AAA Battery option |

* System Status LED
  + LED will be OFF by default
  + LED will only come on if:
    - Unable to read sensors
    - Battery needs charging
    - Cannot connect to Particle
    - Cannot connect to Blynk
  + LED may be multicolor or flash different pattern depending on error – TBD
  + LED is powered independently of connectivity so it will light up even if system is in error and cannot connect. Will remain on after triggered even if system goes to sleep.
* Battery architecture is modified so that the system battery becomes the primary power source and the reserve battery is only useful as a backup if the system battery is not changed in time or while the battery is being changed.
  + Send a notification when on reserve battery?
  + Battery charge is based on the fuel gauge for system battery
  + When system battery is disconnected or drained the battery charge percentage is based on the reserve battery fuel gauge
* Confirm performance of sensors with new design – may not change much!