ED1608 Application note #002

Tilt measurement for alarm purposes

TILT sensor explained

The ED16XX libraries contain a lot of functions.
This application note explains the use of the STR subfunction for tilt detection.

The TiltDetector is a function that uses XYZ values from the accelerometer and calculates the angle in degrees between the gravity vector and a reference vector. The angle has no direction, but the calculation will work for any default orientation.
The accelerometer is active at a sampling frequency (ODR) of 50 Hz and updates the GravX, GravY and GravZ values.

Library H-file definitions:

```
typedef struct {                        //
    unsigned int Enabled     :1;        // enable or disable the calculation of GravityAngle
    unsigned int QuickAdapt  :1;        // [0,1] 0 = once per second,
                                        //       1 = every 50 ms while not stable
  } TTiltDetectorConfig;                //

TTiltDetectorCfg TiltDetectorCfg;       // Configuration of Tilt Detector. Default all 0

int16 GravityAngle;                     // Calculated angle in degrees between current
                                        // gravity vector and reference gravity vector
                                        // Unit is in 0.1 degrees, horizontal = 0.0 degrees.
                                        // Example: A value of 100 means 10.0 degrees.
                                        // Range 0 to 900 units (0.0 to 90.0 degrees)


void  CalibrateRefPlane();              // reset the reference vector to the current value
void  Init_AccRefPlane();               // called by library on boot for module initialization
```

Explanation:
The `TiltDetectorCfg.Enabled` bit enables or disables the calculation. If not needed, it is wise to disable the calculation to save computing time and therefore power. It takes about 284 us to calculate the angle.
When this algorithm is run every second this will use 284 us additional computation time = CPU activity.
CPU current = 23.6 mA, time is 284 us so daily use in mAh will be 284/1000000 *23.6 *24 = 0.161 mAh per day
This is not really significant, in a situation where a 6000 mAh battery would last 583 days this will change to 574 days.

The `TiltDetectorCfg.Enabled` bit enables or disables temporary short update intervals when large changes in tilt are detected. Normally new values are low pass filtered every second, but for large changes it takes a lot of time to get a stable value. When the interval is 10 times faster a new stable value is also 10 times faster, at the expense of keeping the CPU and timers awake (more power).

The int16 variable `GravityAngle` only tells the tilt angle in degrees with respect to the horizontal plane. If any information about the direction of the tilt is required an analysis of GravX, GravY and GravZ is required.

The `CalibrateRefPlane()` function will setup a mathematical plane that is perpendicular (NL: loodrecht) to the actual gravity vector at the moment this function is called. This plane represents a horizontal plane that is parallel to the Earths horizon, and is used to calculate the angle between the actual gravity vector and the calibrated vector.

It does not matter in which orientation the device is when this function is called. From then on, the calibrated orientation is considered horizontal.

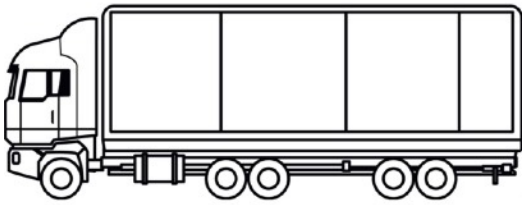
The `Init_AccRefPlane()` function is called from the library startup.
There is no need for the application to call this function.

A horizontal orientation produces a result of 0.0 degrees with about 0.1 degree accuracy.
Accuracy will degrade when the sensor is vibrating or shaken. The results will vary a little over time.

Example: Tire theft alarm

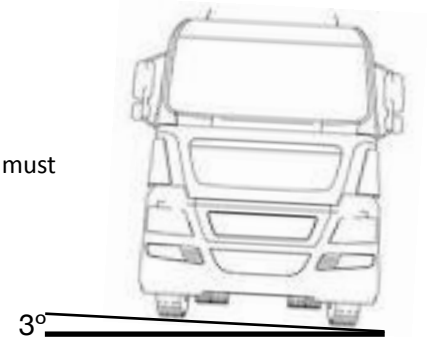
Tire theft alarm on trucks and trailers. For large trailers it will be almost impossible to measure the tilt in the length of the truck, but since most trucks are around 2.5m wide this is the movement we will use.

A few assumptions:



1. When a tire is stolen the truck or trailer is stationary and has been stationary for a while.
2. To steal a tire, the truck is lifted at least 10 cm.
3. When a vehicle is parked, deviations to horizontal are easily over 10 cm.
4. Parked or driving: wind and acceleration forces are well over the 10 cm threshold.

This requires a number of things:



1. While driving, the tilt detection should be disabled.
2. The tilt vector requires heavy low pass filtering.
3. When the vehicle is stationary and stable for a while, the system must be nulled and enabled.

When a truck is 2.5 meters wide and lifted for 15 cm, the angle will be inverse sine(0.15/2.5) = 3.4 degrees.
When a truck is 2.5 meters wide and lifted for 10 cm, the angle will be inverse sine(0.10/2.5) = 2.3 degrees.
When the trigger value is set to 1.0 degrees, the tilt matches 0.017*2.5 = 4.3 cm.

Problems

- One of the problems is how to prevent a false alarm when the truck starts driving normally. To prevent this false alarm the software has “drive” detection, this detection takes at least 2 minutes of moving to change its state.
- Another is a change in tilt as a result of (un)loading cargo or using a ferry. In this case the drive detection will not be active, so false alarms will be triggered. One way to stop this is to have an ignition key detection.

Downlink parameters

The information for downlinks can also be found in the "Technical product description" of the ED1608 tracker.
Commands for setting configuration parameters and System Commands follow a specific format which has a valid 2 byte security checksum.
The parameter packet uses the following struct:

```
typedef struct {
        byte ParamID;       // Param/Cmd identifier
        word ParamValue;    // 2 byte value
        byte CfgSequence;   // 1 byte sequence for confirmation in Alive Message
        word CS;            // 2 byte security check
} TCmdStruct;
```

Tilt downlink:

ParamID : 0x36
ParamValue: 0=off, 1..22 =angle (in degrees)
CfgSequence: 0xFF
CS: 0x6893 (16 bits CRC checksum)

Final downlink: 360005FF6893

On our website we have a downlink calculator, https://www.1m2m.eu/webtools.php

PayLoad description

Parameters:

1. Movement sensitivity. (normally set between 5 to 15)
2. Alarm tilt angle
3. Time the system has not detected movements before nulling and arming the tilt sensor alarm.
4. Time that low-pass filtered tilt angle must be above threshold before an alarm is sent.
5. Low pass filter coefficient

The payloads can be decrypted via the 1M2M Payload decoding JSON service
https://1m2m.eu/services/GETPAYLOAD?Human=0&PL=YourPayload

There are web tools for coding and decoding available on https://www.1m2m.eu/webtools.php