Substituting GPS with Gyroscope and Accelerometer

Jakub Šťastný
Student number: 03729788
Course: Real-time Systems [IN2060]

Abstract

Submit your project report (5 to 10 pages per person) by the 4tth of February 2021 (channel will be announced later). The report needs to contain the names of the authors, which course are they on (IN2060 or IN8014) and their student number (matrikel number). You must also state that you equally contributed to the project if you worked in pairs. Before you do this, use the form below to register your project (very brief description required only) via moodle. I will open a submission channel for this. The submission of your project registration has to take place before by the 17th of December 2020. You need to be prepared to give a talk about your project on the 11th of February 20201. This will happen during our normal lecture time. I will randomly select several projects on the day of the presentation, so you all have to be prepared to present You will receive a 0.3 grade bonus on your exam (independent of the quality of you presentation) if you hand in a reasonable project report (you still have to pass your exam with a grade ¡=4.0). If you pass the exam with a 1.0 grade, the project will not further improve your grade. There is no grade for your project, only a pass and fail which means you will either get the exam bonus or not.

1 Project description

In this project I have built the device that shows to the user his/her current position and speed on the display in a real time. In normal case, the device is using only a GPS chip in order to get the information about the current location. But there are many cases, when we loose a GPS signal. For example inside of some buildings, between high mountains, in tunnels, etc. In these cases we want to have the device still running so we have to substitute the GPS with some another alternative that is independent on the quality of any signal. I decided for the gyroscope and accelerometer that can measure our acceleration and angular speed, which can be using double integration converted to the relative location[1] and combined with the last location that the GPS measured before it lost signal, in order to calculate the current location. With this technique we can keep the device running and providing the output to the user for the whole time even if we do not have any signal.

Besides the controller board, accelerometer, gyroscope and display, the device contains two additional buttons. One of them is used for the (re)calibration of the device (described later). The second button is present only for the testing purpose and is used for the simulation of the situation that we do not have any GPS signal. If a user presses this button, he/she turns on the testing mode so that the device does not use any information from the GPS, even if it has a GPS signal and calculates the current speed and location only using the accelerometer and gyroscope data in order to simulate and test the situation that we do not have a GPS signal. If the user presses the button again, the testing mode is turned off and the device is working normally.

The device has also one additional LED, which indicates that the testing mode is turned on.

2 Used hardware

For this project I used following hardware:

1. LILYGO TTGO T-Beam V1.1[2]
2. MPU-6050 3 Axis Gyro With Accelerometer[3]
3. OLED 128x64 I2C SSD1315 12864 LCD Screen Display module[4]
4. 2 Buttons, 1 LED and 18650 Battery

2.1 Controller board

As a microcontroller for this project I used LILYGO TTGO T-Beam board version 1.1. This board is based on ESP32 processor. It offers 4 MB Flash memory and 4 MB of PSRAM. The board also contains 2 additional modules. First of them is module LoRa32 which is used for wide area communication via bluetooth and WiFi. I haven't used this module for the project. The second and for the project more important module is the NEO-6M module used for GPS. This module can measure current gps location with the accuracy up to 2.5 meters[5]. This module communicates with the ESP processor via integrated serial BUS in format of NMEA sentences[6].

2.2 Gyroscope and Accelerometer

The MPU-6050 is a 3 axis gyroscope and accelerometer, which can communicate via I^2C protocol. The accelerometer can measure the acceleration in the x, y, and z axis[7]. With the configuration

[1] The location difference between our current location and location before the start of the acceleration measurement

[2] Detailed description: `https://www.banggood.com/LILYGO-TTGO-T-Beam-ESP32-433-or-868-or-915-or-923Mhz-V1_1-WiFi-Wireless-bluetooth-Module-GPS-NEO-6M-SMA-LORA32-18650-Battery-Holder-With-OLED-p-1545070.html`

[3] Detailed description: `https://www.banggood.com/3pcs-6DOF-MPU-6050-3-Axis-Gyro-With-Accelerometer-Sensor-Controller-Modu html`

[4] Detailed description: `https://www.amazon.com/Serial-Display-Module-SSD1315-arduino/dp/B07Y5HNJGL`

[5] This accuracy can be achieved only with a quite good antenna or a clear sky view. With the antenna I used for the project is the accuracy between 15 and 20 meters.

[6] Described here: `http://aprs.gids.nl/nmea/`

[7] The axis are defined relatively to the accelerometer and not in earth coordinate system

I used to the project, the accelerometer can measure in each axe the acceleration in range from -2 g to 2 g with the sensitivity $6.1 \cdot 10^{-5}$ g. The gyroscope can measure the angular speed of a board around each x, y and z axe. It can be measured in range from -250 $^\circ s^{-1}$ to 250 $^\circ s^{-1}$ with the sensitivity $7.6 \cdot 10^{-3}$ $^\circ s^{-1}$.

2.3 Display

To display current speed and location to the user I used OLED 128x64 12864 LCD Screen Display. This display communicates again via I^2C protocol. The screen is 128 pixels wide and 64 pixels high.

2.4 Circuit

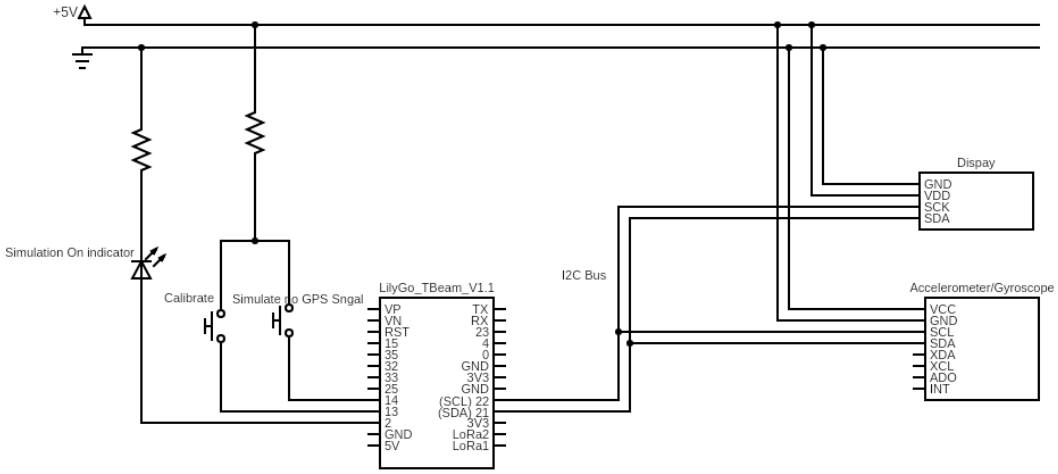

Figure 1: Circuit scheme (power supply of the TBeam board and power lines connection left out)

3 Reading and processing accelerometer and gyroscope data

The main challenge of the processing the accelerometer data is the fact that we receive the acceleration relatively to the accelerometer module and not relatively to the earth coordinate system and we need to convert them so we can calculate the final location. Furthermore for the conversion we need an orientation of the board relatively to the earth coordinates which is not so easy to get without a tilt sensor or something similar.

3.1 Processing and converting raw data

As mentioned earlier we obtain from the accelerometer data relatively to the module axis. From the accelerometer and gyroscope we receive three 16-bit integers that correspond to the acceleration in x, y and z axes multiplied by 16384 in multiplies of g. After these 3 numbers follow three another 16-bit integers that correspond to the angular speed around x, y and z axes multiplied by 131 in degrees per second.

For this part let's assume that we know the exact orientation of the module relatively to the earth coordinate system. If we know the orientation we can easily convert received acceleration data to the earth coordinates system that we convert received acceleration vector to the polar coordinates system, than we can easily rotate the vector by the orientation vector and than convert back to the cartesian coordinates system and we obtain a vector of accelerations in x, y and z axe relatively to the earth coordinates system.

3.2 Device calibration

In this section I will talk about the way how can be obtained the orientation of the accelerometer module relatively to the earth coordinates system using only gyroscope and accelerometer. The calibration is a process that starts after user presses the calibrate button and takes ca 10 seconds. During the calibration the device can be rotated as desired but it must not move for the whole time. The main goal of the calibration is to obtain the orientation of the module relatively to the earth coordinates system. During the calibration we also calibrate the gyroscope so that it measures correct data.

If the device is in calm, we expect the acceleration 0 in x and y axes and -1 g in z axe[8] if the module coordinates would be the same as the earth coordinates. During the calibration time we measure the current acceleration. In order to minimize the error of the measurement, we measure the acceleration multiple times and after we have enough measurements we can calculate average value and use it for the further calculation. After theses measurements we have an information about the calm acceleration relatively to the module which is usually different from the expected $[0, 0, -1]$ g vector. At this point we can convert both vectors to the polar coordinates system and calculate the rotation differences between both vectors. In other words we can calculate how should we rotate the measured vector so that we obtain the expected vector, which is the information about the module orientation that we need for the conversion to the earth coordinates system.

At this point we know the initial orientation of the device. But this orientation can change during the usage of the device and changes quite often. To handle this, we can use the data from the gyroscope. The gyroscope measures the angular speed of the module, so if we detect that there is some angular speed around any axe different from 0, we know that user tilted the device and we can accordingly adjust our information about the orientation. Updating the orientation is the only part for which I am using the gyroscope in the project.

The last thing I am doing during the calibration process is the calibration of the gyroscope. Because of the construction error almost each gyroscope has shifted the 0 levels of the angular speeds. During the calibration process I can measure the also the gyroscope values (and calculate their average). Because the device is in calm, the measured values should be all 0, but because of the shift error, each value is shifted so we obtain the error shift values that we can further subtract from the measured values in order to get right angular speeds.

3.3 Data smoothing

From this point we know, how to recalculate the measured values to the earth coordinates system so we will assume that all accelerations are in the earth coordinates system.

In this section I will talk about the preparation of the measured data before we can start the final computation of the speed and location. The problem, why we shouldn't directly calculate speed and location from the measured date, is that because of the double integration, even small measurement errors can have huge consequences and we have to minimize these errors as much as possible. If we observe the values from the accelerometer, we can see that they are "oscillating" around a real acceleration[9] (See figures 2 and 3). This phenomenon could be partially solved with smoothing.

Now we assume that we have measured some data, let's say n records and we want to smooth them. Let's choose a value b and divide the set of measured records into $\frac{n}{b}$ blocks of b consecutive records. In each block we will find a median value of each x, y and z axe and set all records in the block to this median. This will cause that we will avoid or smooth these oscillations or a noise in form of several random completely wrong records.

This smoothing technique is really simple, fast and reduces the errors quite well. The disadvantage of the smoothing is, that we partially loose precision level of the measurement. For example, if there would be some really short acceleration that takes only short time, it could be assumed as an error and completely ignored. Or if we have block where the acceleration is changing, we would loose the information about the shape of the curve, hot the acceleration is changing and we assume it later as linear.

[8] This is caused by gravity, the $[0, 0, 0]$ acceleration would be measured if the module will be in a free fall

[9] This can be easily observed, if the device is in calm and the acceleration oscillates randomly around 0

The level how much the data will be smoothed and how large the information lost is depends on the value of b. If we choose very small b, in extreme case $b = 1$, we wouldn't loose almost any information but we will smooth really small blocks so the smoothing wouldn't have almost any effect and most of the errors will remain. On the other hand, if we choose really large b, in extreme case $b = n$, we will ignore almost all errors, but also almost every information we obtain, so the data would become almost useless.

In this project, I chose $b = 16$ records[10]. It is quite small, so we wouldn't loose almost any information, because the change of the acceleration is in reality not so fast that the curves inside each block can be almost every time quite good approximated to the linear values. But it is not too small and it smooths data quite well, especially if the device is not accelerating at all.

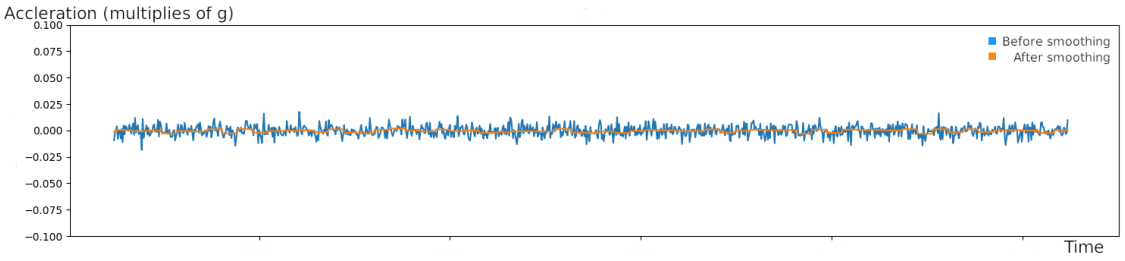

Figure 2: Acceleration on the x axe before and after smoothing when the device is in calm

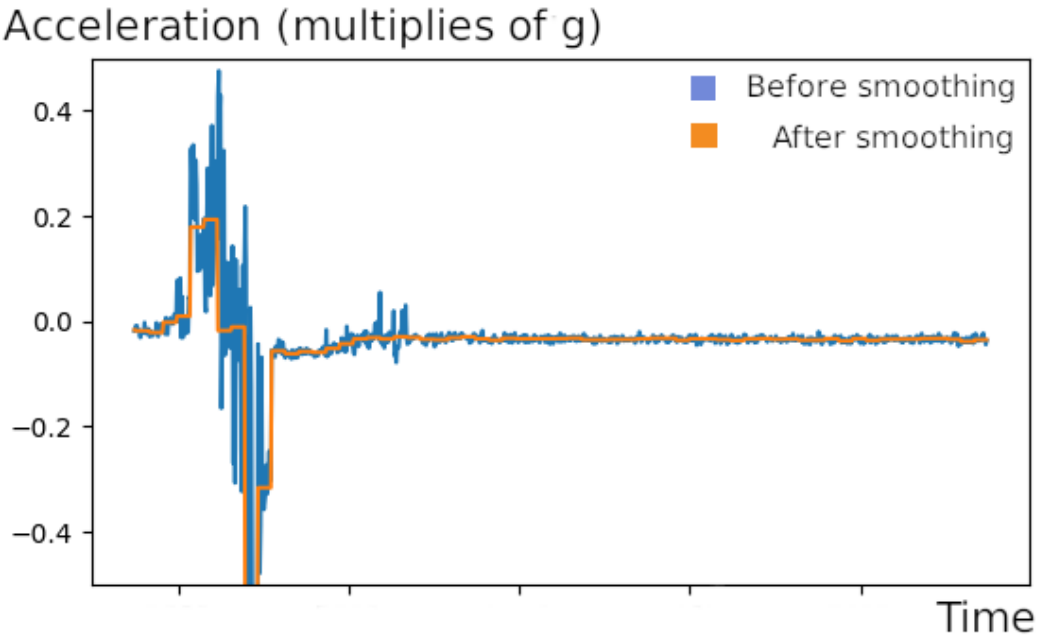

Figure 3: Acceleration on the x axe before and after smoothing when the device is in motion

After the data are smoothed, we could apply another technique to reduce an error. Even after smoothing, if the device does not accelerate, we can still get some small positive or negative accelerations, which need to be removed. To do that we can choose some limit l and than assume every acceleration, which is in the absolute value lower than the limit, as 0. For the project, I chose $l = 0.03125\ g$.

[10] And I am working with data sets with $n = 512$ records, so I divide them into 32 blocks. More about it in the next section.

3.4 Calculating final speed and location

Now we have reduced the measurement error and we can finally calculate the final location. Let's assume, we know the initial speed v_0 and location p_0 of the device[11] and set of n last acceleration measurements together with the times of the measurement.

To calculate a velocity and a position from the acceleration we need to convert the discrete values into a continuous function. Because the time differences between measurements are quite small, compared to expected accelerations, we can assume, that before each 2 measurements is the acceleration constant and approximate our discrete function to the continuous one in the way displayed on the Figure 4.

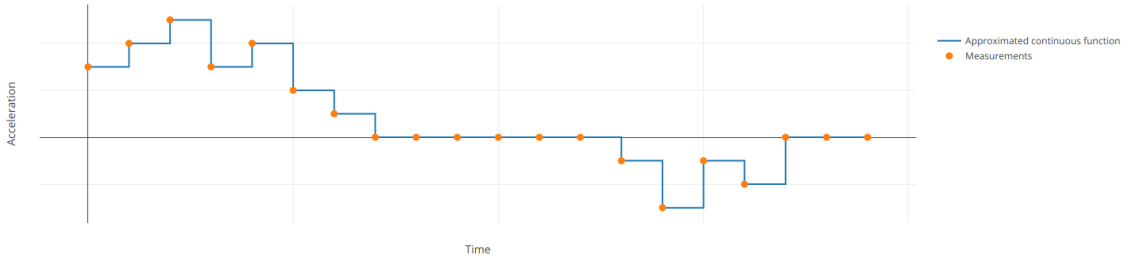

Figure 4: Approximating discrete measurements to continuous function

Now, we have all information we need, to calculate a velocity as

$$v(t) = v_0 + \int_{T_{\text{start}}}^{t} a(\tau)\ d\tau$$

Where v_0 is initial speed, T_{start} is a time of the first record in measurements set, T_{end} is a time of the last record and $a(t)$ is the approximated acceleration function from the measurements. The value $v(T_{\text{end}})$ will be final speed used as initial speed for the next measurement.

When we know also the function of velocity with respect to time, we can integrate it again to obtain our relative position with respect to the last know position.

$$p_r = \int_{T_{\text{start}}}^{T_{\text{end}}} v(t)\ dt$$

Now we have a relative position p_r so we can convert it from meters to longitude and latitude degrees and add it to the last known location p_0 and obtain our current location.

The last thing we have to do is to choose appropriate number of records n that we will collect before we recalculate the location. If we choose n relatively small, we will recalculate the position quite often, so the user will see the information about the position always up to date. If we choose n quite large, it will take some time before we collect enough data to recalculate the location and user would have to wait some time before he/she sees the new updated values. But on the other hand, the larger n we choose the more data we have to calculate and we can perform more appropriate error correction, function approximation and we have more also denser data, because we only collect them and we do not always recalculate the position, so we can obtain more accurate values.

In the project, I chose $n = 512$. We have quite enough data to do appropriate error corrections and collecting + recalculating takes a bit less than one second, so the data is almost always up to date[12].

[11] The initial speed will be 0 after calibration or result of the last calculation. The initial position will be last record from GPS, before we lost the signal or result of the last calculation

[12] The target user is a human not another machine, so updating state in less than one second is fast enough

3.5 Position recalculating pseudocode

If I sum up all previous steps into pseudocode, it will look like this.

1. Let v_0 and p_0 be initial speed and location
2. measure data until you have n records
3. smooth data
4. $v :=$ calculate speed from smoothed data
5. $p :=$ calculate relative position from the smoothed data
6. $p' :=$ convert relative position to the earth the coordinates units
7. $p_0 := p_0 + p'$
8. $v_0 := v$
9. delete measured data
10. goto step 2

4 Results and precision

4.1 Accelerometer inaccuracy

5 Conclusion