3D Motion tracking with Gyroscope and Accelerometer

Final Project

By:
Shay Reuveny
Maayan Zadik

Supervisor:
Gaddi Blumrosen
(Boris Rubinsky)

Table of Contents

1. Introduction

In our project we use an existing "Body Sensors Network" system called BSNv3, which contains a CPU, a radio transceiver and a 3D accelerometer.
In order to correct the results given by the accelerometer, since an accelerometer cannot detect motion with constant speed or turns, and to receive additional information about the motion in the 3D space, we add two dual-axis gyroscopes, which give information about all 3 axes.

Figure 1.1 *BSN Kit*

A wireless module, containing all the sensors and mounted upon the subject, will send the information to another, remote, module which is connected to a PC, using the radio transceiver of the BSN node.
The information sent from the accelerometer and the gyroscopes will be combined in real-time using an algorithm which we will choose and implement, and will give us the overall info about the motion.

The analyzed data will be displayed graphically on a GUI, in order to make the data readable and understandable.

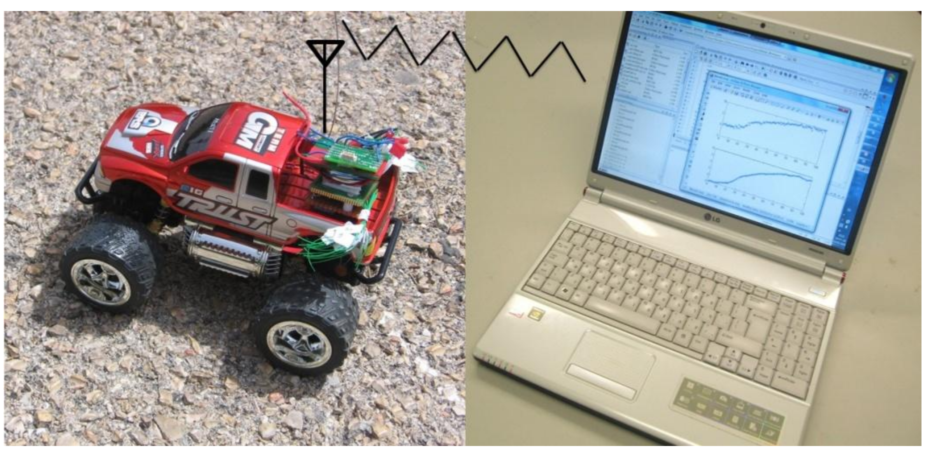


Figure 1.2 *An illustration of the module sending data to the remote computer*

Research goals:

Our goal is to assess continuous motion with a 3D accelerometer and a combined 3D gyroscope, and make it as accurate as possible.
This will be of future use in medical or educational application, such as determining the severity of Parkinson's disease, monitoring the healing process of a damaged limb, or detecting a "proper" movement in sports.

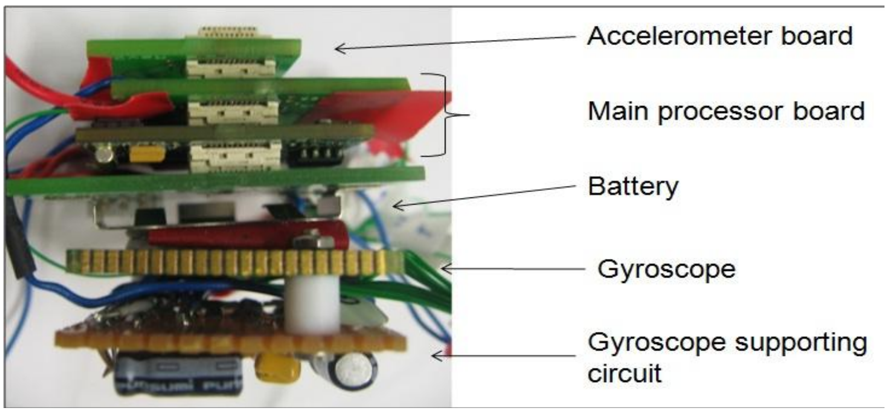


Figure 1.3 *The complete module*

2. System model

2.1 System components

Our system consists of four modules:

1. One remote wireless BSNv3 nodes, with a CPU and a radio transceiver.

2. A 3D accelerometer mounted on a BSN sensor board and connected to the BSN node.

3. Two dual-axis gyroscopes, XY and XZ, mounted on an additional 'prototype' sensor board supplied with the BSNv3 kit, and also connected to the BSN node.

4. A Telos-B module, which is connected to a PC and receives the data sent by the remote BSN node, sending it to the computer for further analysis.

2.2 Accelerometer principles

As is his name, accelerometer measures acceleration, a.

$$a = \left(a_{x,}, a_y, a_z\right) \tag{2.1}$$

Accelerometers can measure in one, two, or three orthogonal axes. They are typically used in one of three modes:

- As an inertial measurement of velocity and position
- As a sensor of inclination, tilt, or orientation in 2 or 3 dimensions, as referenced from the acceleration of gravity (1 g = 9.8m/s^2);
- As a vibration or impact (shock) sensor.

The most common acceleration is as a result of the earth's gravitational pull. Since it's difficult to measure acceleration directly, the accelerometer measures the force exerted by restraints placed on a reference mass to hold its position fixed in an accelerating body.

When looking for and working with an accelerometer it's important to know its type of output. In both analog and digital accelerometer the output value is a scalar corresponding to the magnitude of the acceleration vector.

Analog accelerometers output a constant variable voltage depending on the amount of acceleration applied. The signal can be a positive or negative voltage, depending on the direction of the acceleration.

Since the output of this type of accelerometer will be fed to an ADC, the value must be scaled and/or amplified to maximally span the range of acquisition.

Digital accelerometers output a variable frequency square wave, a method known as pulse-width modulation (PWM). A PWM accelerometer takes readings at a fixed rate.

Accelerometers with PWM output can be used in two different ways. For most accurate results, the PWM signal can be input directly to a microcontroller where the duty cycle is read in firmware and translated into a scaled acceleration value. When a microcontroller with PWM input is not available, a simple RC reconstruction filter can be used to obtain an analog voltage proportional to the acceleration. These voltages can then be scaled and used as one might the output voltage of an analog output accelerometer.

2.3 Gyroscope principles

Gyroscope measures angular velocity, w. An angular rate gyroscope is a device that produces a positive-going output voltage for counter-clockwise rotation around the sensitive axis considered.

2.3.1 Introduction

A gyroscope is a device used primarily for navigation and measurement of angular velocity expressed in Cartesian coordinates. It produces a positive-going output voltage for counter-clockwise rotation around the sensitive axis considered. There are gyroscopes available that can measure rotational velocity in 1, 2, or 3 directions. A 3-axis gyroscope combined with a 3-axis accelerometer provides a full 6 degree-of-freedom (DoF) motion tracking system.

2.3.2 Terms

Vibrating Structure Gyroscopes are MEMS (Micro-machined Electro-Mechanical Systems) devices that are easily available commercially, affordable, and very small in size.

Gyroscope Sensitivity describes the gain of the sensor and can be determined by applying a defined angular velocity to it.

Zero-rate level describes the actual output signal if there is no angular rate present. The zero-rate level of precise MEMS sensors is, to some extent, a result of stress to the sensor and therefore zero-rate level can slightly change after mounting the sensor onto a printed circuit board or after exposing it to extensive mechanical stress.

Measurement range specifies the maximum angular speed with which the sensor can measure, and is typically in degrees per second (˚/sec).

Number of sensing axes is the number of axes in which the gyroscope measures the angular rotation. Gyroscopes are available that measure either in one, two, or three axes. Multi-axis sensing gyros have multiple single-axis gyros oriented orthogonal to one another.

Nonlinearity is a measure of how close to linear the outputted voltage is proportional to the actual angular rate. Not considering the nonlinearity of a gyro can result in some error in measurement. Nonlinearity is measured as a percentage error from a linear fit over the full-scale range, or an error in parts per million (ppm).

Working temperature range specifies the temperature range in which the gyroscope works correctly and fails beyond that.

Shock survivability specifies how much force the gyroscope can withstand before failing. Fortunately gyroscopes are very robust, and can withstand a very large shock (over a very short duration) without breaking. This is typically measured in g's (1g = earth's acceleration due to gravity), and occasionally the time with which the maximum g-force can be applied before the unit fails is also given.

Bandwidth typically measures how many measurements can be made per second. Thus the gyroscope bandwidth is usually quoted in Hz.

Bias is the signal output from the gyro when it is NOT experiencing any rotation. Bias can be expressed as a voltage or a percentage of full scale output, but essentially it represents a rotational velocity (in degrees per second). Unfortunately bias error tends to vary, both with temperature and over time. The bias error of a gyro is due to a number of components: Individual measurements of bias are also affected by noise, which is why a meaningful bias measurement is always an averaged series of measurements.

Bias Drift refers to the variation of the bias over time, assuming all other factors remain constant. This effect is caused by the self heating of the gyroscope and its associated mechanical and electrical components. This effect would be expected to be more prevalent over the first few seconds after switch-on and to be almost non-existent after about five minutes.

Bias Instability is a fundamental measure of the 'goodness' of a gyroscope. It is defined as the minimum point on the Allan Variance curve, usually measured in °/hr. It represents the best bias stability that could be achieved for a given gyro, assuming that bias averaging takes place at the interval defined at the Allan Variance minimum.

2.3.3 The physical principles of vibrating structure gyroscope

Fundamental to an understanding of the operation of a vibrating structure gyroscope is an understanding of the Coriolis force. In a rotating system, every point rotates with the same rotational velocity. As one approaches the axis of rotation of the system, the rotational velocity remains the same, but the speed in the direction perpendicular to the axis of rotation decreases. Thus, in order to travel in a straight line towards or away from the axis of rotation while on a rotating system, lateral speed must be either increased or decreased in order to maintain the same relative angular position (longitude) on the body.

$$F_{coriolis} = m \cdot a_{angular} \qquad (2.2)$$

Where $a_{angular}$ is angular acceleration and m is the mass of the object whose longitude is to be maintained

The Coriolis force is proportional to both the angular velocity of the rotating object and the velocity of the object moving towards or away from the axis of rotation.

2.3.4 The mechanics of vibrating structure gyroscope

Vibrating structure gyroscopes contain a micro-machined mass which is connected to an outer housing by a set of springs. This outer housing is connected to the fixed circuit board by a second set of orthogonal springs.

The mass is continuously driven sinusoidally along the first set of springs. Any rotation of the system will induce Coriolis acceleration in the mass, pushing it in the direction of the second set of springs. As the mass is driven away from the axis of rotation, the mass will be pushed perpendicularly in one direction, and as it is driven back toward the axis of rotation, it will be pushed in the opposite direction, due to the Coriolis force acting on the mass.

The Coriolis force is detected by capacitive sense fingers that are along the mass housing and the rigid structure. As the mass is pushed by the Coriolis force, a differential capacitance will be detected as the sensing fingers are brought closer together. When the mass is pushed in the opposite direction, different sets of sense fingers are brought closer together; thus the sensor can detect both the magnitude and direction of the angular velocity of the system.

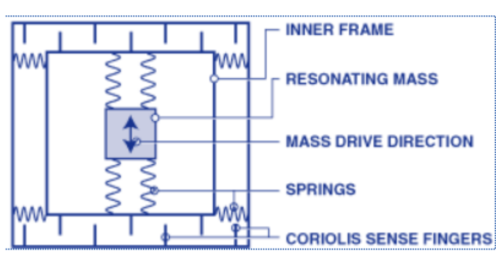


Figure 2.1 *A single-axis vibrating structure gyroscope schema*

The first problem pending was which gyroscope to choose, since the BSN kit had 3 interfaces to which we could attach extra sensors.

2.3.2 Gyroscope mathematical model

A gyroscope is a device used primarily for navigation and measurement of angular velocity expressed in Cartesian coordinates:

$$w = \left(w_{x,}, w_y, w_z\right) \quad (2.3)$$

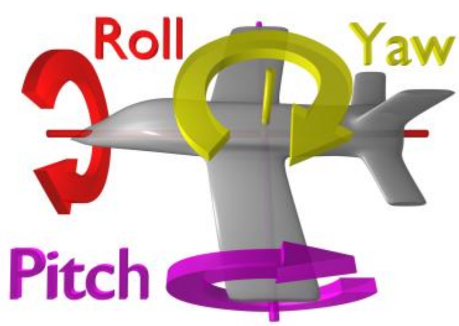
It produces a positive-going output voltage for counter-clockwise rotation around the sensitive axis considered. The coordinates are of Inertial Axes (or Body Axes) - based on Center of Gravity (CG)

The body axes positions are:

- X Axis - Positive forward, through the nose of the object
- Y Axis - Positive to Right of X Axis, perpendicular to X Axis
- Z Axis - Positive downwards, perpendicular to X-Y plane

The Angles definitions are:

- Pitch - Angle of X Axis relative to horizon. Also a positive rotation about Y Body Axis
- Roll - Angle of Y Axis relative to horizon. Also a positive rotation about X Body Axis
- Yaw - Angle of X Axis (nose) relative to North. Also a positive rotation about Z Body axis



22.2 *The angular rotation of an airoplain*

The angle can be derived by integration of the angular velocity in each direction:

$$\theta_p(t) = \int_{t0}^{t} w_p(\tau)d\tau + \theta_{p0} \tag{2.4}$$

Where p is index which can be (x,y,z), Pitch is θ_y , Roll is θ_x and Yaw is θ_z and θ_{p0} is the initial angle compared to the earth axis.

In many times we need to work with earth Axis coordinates which are defined as follow:

- X Axis - Positive in the direction of North
- Y Axis - Positive in the direction of East (perpendicular to X Axis)
- Z Axis - Positive towards the centre of Earth (perpendicular to X-Y Plane)

In order to convert from body axes to earth axes we use the following matrix:

$$\begin{bmatrix} \cos(\theta_z)\cdot\cos(\theta_y) & -\sin(\theta_z)\cdot\cos(\theta_x)+\cos(\theta_z)\cdot\sin(\theta_y)\cdot\sin(\theta_x) & \sin(\theta_z)\cdot\sin(\theta_x)+\cos(\theta_z)\cdot\sin(\theta_y)\cdot\cos(\theta_x) \\ \sin(\theta_z)\cdot\cos(\theta_y) & \cos(\theta_z)\cdot\cos(\theta_x)+\sin(\theta_y)\cdot\sin(\theta_x) & -\cos(\theta_z)\cdot\sin(\theta_x)+\sin(\theta_z)\cdot\sin(\theta_y)\cdot\cos(\theta_x) \\ -\sin(\theta_y) & \cos(\theta_y)\cdot\sin(\theta_x) & \cos(\theta_y)\cdot\cos(\theta_x) \end{bmatrix}$$

Figure 2.3 *The angular rotation of an airoplain*

Where θ_x is the roll around the x-axis, θ_y is the pitch around the y-axis and θ_z is the yaw around the z-axis, as shown below:

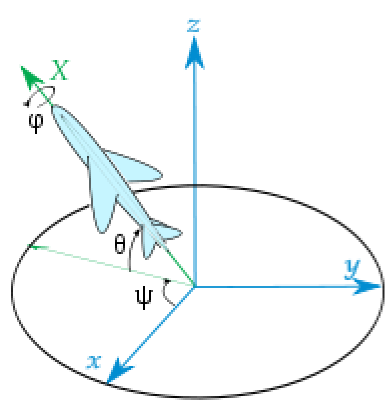


Figure 2.4 *Body to earth example*

The matrix is used to convert the angles from angles referring to the body's movement from its previous position, to angles referring to its movement from the position before it. When it is then multiplied with all the previous matrixes, since each matrix represents one movement, the multiplied outcome represents all the movements since the beginning of the measurements, thus multiplying the body axes coordinates, [Xb, Yb, Zb], with the above mentioned matrix, will result in the object's coordinates in earth axes.

3. Implementation

3.1 HW Implementation

3.1.1 BSN node

BSN node consists of a CPU, A flash memory and a radio transceiver.

In order to connect to additional sensors it has analog and digital interfaces:

Analog interface

1. An analog interface – The CPU on the main board has several ADC pins to which analog devices can be connected.
2. It has 8 ADC's (ADC0 to ADC7) to connect the analog sensor output, together with 'SESOR_PWR' and 'GND' pins that can supply the power and the ground for the sensor.
3. As for the actual connection, it can be made by extra 2 ADC's (ADC4 to ADC5) on the board containing the accelerometer, or by the "Prototype" board available with the kit,

Digital interface

1. A Digital I2C Interface – An I2C interface is a serial bus interface that uses the I2C protocol. A sensor that uses that interface can be connected to the kit's 'Prototype' board through the 'SENSOR_SCL' and 'SENSOR_SDA' pins that connect to the I2C clock signal and to the data signal respectively, and also to the power and ground pins as the analog sensors.
2. A Digital UART/RS232 Interface – A sensor with a UART interface can be connected to the 'UART1TX' and 'UART1RX' pins on the 'Prototype' board, or by a BSN Serial interface board, that is supplied separately from the kit.

	Digital	Analog
Pros	• Synchronized, own power regulations.	• Simple, operation same as accelerometer. • Prototype board has 8 available ADC pins – can connect more than one gyroscope. • More sensitivity and range choices.
Cons	• Different usage from the accelerometer - which is analog • There is only one set of I2C/UART clock and data pins – we would need another board to use 2 gyroscopes. • Not widely available.	• Not synchronized. • Have to regulate power supply ourselves. • Have to translate results to valid data.

3.1 *Digital VS. Analog gyroscope*

3.1.2 Types of Gyroscope

From the various websites and companies we found that supply gyroscopes or full motion sensors containing a gyroscope, the three most prominent ones that were suitable for our needs were the "Analog-Devices", "Invensence" and "STMicroelectronics" company websites.
3-D gyroscope is not available yet.

1. 'Analog Devices' Gyros
   http://www.analog.com/en/mems/gyroscopes/products/index.html

The Analog-Devices company supplies gyroscopes with an analog or digital interfaces, but almost all of them are 1-axis gyroscopes, and since we were looking for a tri-axis one, is was not good for us. This company did have a tri-axis gyroscope, but it was a part of a kit that costs about 320$, so we looked for other solutions.

2. 'Invensence' Gyros
   http://www.invensense.com/store/index.html

The Invensence company supplies gyroscopes using the I_2C interface, that we can use to connect them to the BSN board, and are also cheap in comparison to other gyroscopes we encountered (around 9$ per piece). Although the online store in the above mentioned website displayed only 1-axis and dual-axis gyroscopes available for purchase, we found an article in another website claiming that the company is about to launch a new tri-axis gyroscope, the ITG-3200 gyro chip, that is the first of its kind and costs only 3$ per piece.
After a short correspondent with the Invensence's sales department, we learned that this chip will only be available on the 3rd quarter of 2010, which is by far too late for our schedule.

We also looked at a digital dual-axis gyroscope, IDG-2000. According to the company site, The IDG-2000 is the first digital gyro with both SPI and I2C interfaces, and is available in Pitch and Roll (X,Y) and Pitch and Yaw (X,Z) configurations.
If we were to use this gyroscope, we would have to figure out how exactly to connect the two separate gyroscopes to the BSN kit. Since the Prototype board has only one I_2C clock and data pins, we were wondering if we can connect both gyros to the same pins in the same Prototype board, or do we actually need one more Prototype board, se we will connect each gyro to a separate board, and use the BSN board's bus behavior to distinguish between the two.
According to the press release on September 8, 2009, the IDG-2000 MEMS gyroscopes began sampling on October 2009 to selected OEM partners. It is not available yet to the public.

We then turned to looking at 2 analog dual-axis gyroscopes, a Pitch and Roll (X,Y) and a Pitch and Yaw (X,Z) gyros.
In order to use two compatible gyros of the same specifications, we could chose either the IDG-500 (X,Y) and the IXZ-500 (X,Z) gyros, both with a sensitivity of 2 mV/°/sec Primary output and 9.1 mV/°/sec Secondary output, and a full scale range of ± 500 °/sec, or the IDG-650 and IXZ-650, with a sensitivity of 0.5 mV/°/sec Primary output and 2.27 mV/°/sec Secondary output, and a full scale range of ± 2000 °/sec.

3. STMicroelectronics Gyros

http://www.st.com/stonline/products/families/sensors/gyroscopes.html

The STMicroelectronics website, which we found recently, provides us with the same solution as the Invensence website, which is to use 2 dual-axis gyros. As opposed to the Invensence gyros, this site offers analog output dual-axis gyros, with broader selection and the same specifications as the Invensence gyros, but no prices were presented on the website. The broader selection may be an advantage to us since we can choose a more appropriate gyro for our needs.

Since we need 2 gyros, we will have to connect both of them to the Prototype board, i.e. to the ADC pins available, together with SENSOR_PER and GND.

These gyroscopes have self-test, ST, and sleep /power down, SLEEP/PD, pins, that we can use to set the gyroscopes to sleep when they are not in use, and using all analog sensors (gyroscopes and accelerometer) might help us to better synchronize the results.

Company	Analog Or Digital	Product Number	Axes	Sensitivity [mV/dps]		Range [dps]		Price [$]	Power [mW] P=V·I	Operation Voltage [V]	Bandwidth [Hz]
				Low	High	High	Low				
InvenSence	Analog	IDG-500	XY	2	9.1	±500	±110	9.00	21	3.0±10%	NA
		IDG-650		0.5	2.27	±2000	±440	9.00	21	3.0±10%	NA
		IXZ-500	XZ	2	9.1	±500	±110	9.00	21	3.0±10%	NA
		IXZ-650		0.5	2.27	±2000	±440	9.00	21	3.0±10%	NA
STM	Analog	LPR403AL/ LPR503AL	XY	8.3	33.3	±120	±30	4.62-6.05	20.4	3.0±10%	140
		LPR410AL/ LPR510AL		2.5	10	±400	±100	4.62-6.05	20.4	3.0±10%	140
		LPR430AL/ LPR530AL		0.83	3.33	±1200	±300	4.62-6.05	20.4	3.0±10%	140
		LPR450AL/ LPR550AL		0.5	2	±2000	±500	4.62-6.05	20.4	3.0±10%	140
		LPR4150AL/ LPR5150AL		0.67	0.167	±6000	±1500	4.62-6.05	20.4	3.0±10%	140
		LPY403AL/ LPY503AL	XZ	8.3	33.3	±120	±30	4.62-6.05	20.4	3.0±10%	140
		LPY410AL/ LPY510AL		2.5	10	±400	±100	4.62-6.05	20.4	3.0±10%	140
		LPY430AL/ LPY530AL		0.83	3.33	±1200	±300	4.62-6.05	20.4	3.0±10%	140
		LPY450AL/ LPY550AL		0.5	2	±2000	±500	4.62-6.05	20.4	3.0±10%	140
		LPY4150AL/ LPY5150AL		0.67	0.167	±6000	±1500	4.62-6.05	20.4	3.0±10%	140

Table 3.2 *Gyroscope comparison and specifications*

The battery given to us by the BSN kit supplies +3.7V and 55mAh. After a quick calculation we concluded that the battery will last almost 10 hours.

From the different kinds of gyroscopes existing, we're going to use a Vibrating Structure Gyroscopes, which are MEMS (Micro-machined Electro-Mechanical Systems) devices that are easily available commercially, affordable, and very small in size.

We chose to order STMicroelectronics' LPR510AL and LPY510AL gyroscopes because they are small, cheaper than InvenSence's and their range/sensitivity is best for our use. Furthermore, if at any time we discover that the range (in amplified mode) is too narrow for us we can always use the wider range (regular mode), giving us room for error and changes in our initial estimate.

We prefer sensitivity over range as our problem does not change in wide angles.

3.1.3 Supporting Circuit Board

In order to connect the gyroscopes to the BSN Kit, we needed to build a supporting circuit board, connect it to the appropriate gyroscope pins and connect them to the BSN prototype board. According to the gyroscope data sheet, there are 3 circuits needed for each gyroscope:

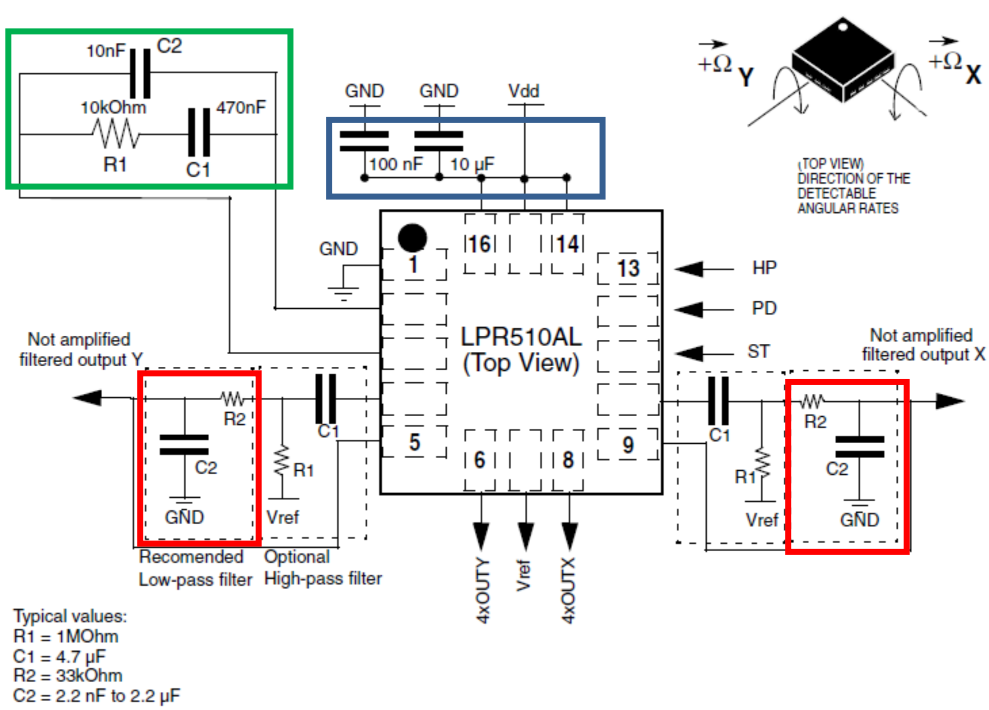


Figure 3.1 *The gyroscope's electrical specifications*

The red circuits, which are identical, are low-pass filters for the X and Y outputs, as displayed in the figure. They connect between the non-amplified output and the amplified input, while reducing the outputted noise. The amplified inputs are internally connected to the amplified outputs, from which we take our results.

The green circuit provides a low-pass filter for the gyroscope's internal synchronization mechanism.

The blue circuit provides voltage stabilization for the voltage supply pin.

Figure 3.2 depicts the board we designed and built, containing all the needed circuits for both gyroscopes. This board, together with the board containing the sensors, connects as a complete gyroscope module to the BSN prototype board.

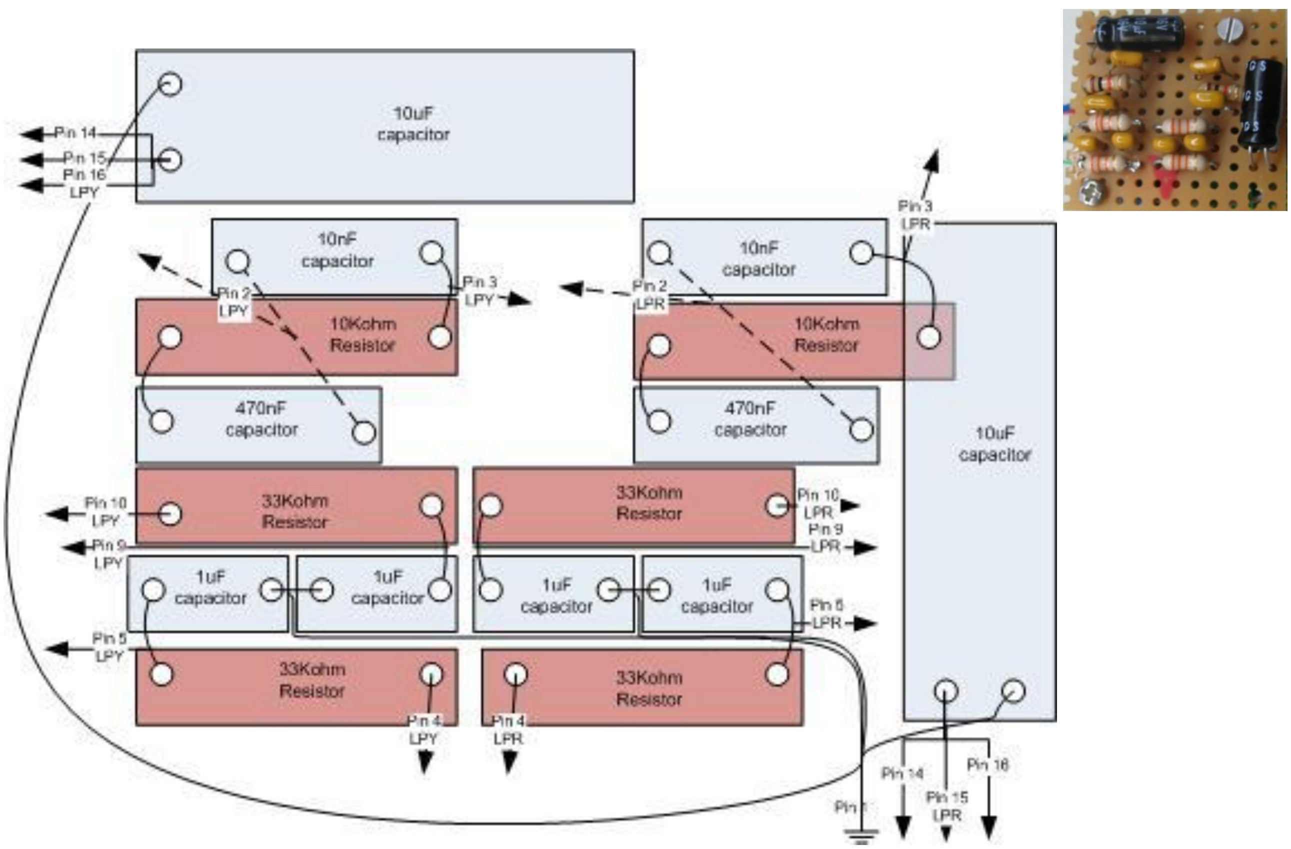


Figure 3.2 *Supporting board design*

3.2 SW implementation

3.2.1 Boot Sequence

The TinyOS platform, that we are using, had a distinct boot sequence that comprises of 3 steps:

1. **INITIALIZE SCHEDULER.**
2. **INITIALIZE PLATFORM AND SOFTWARE.**
3. **SIGNAL THE PROGRAM THAT THE HARDWARE HAS BEEN INITIALIZED.**

This initialization is made in the main module of the TinyOS, which is comprised of a configuration file 'MainC.nc', and a component 'RealMainP.nc', which contains the main() method.

Each platform (e.g. micaz, telos, BSNv3) has a unique implementation of the platform initialization, defined in a configuration module called 'PlatformC.nc', which every new platform that wishes to use TinyOS has to supply.

The Main() method calls a 'PlatformInit.init()' function, that is implemented and wired to the 'PlatformC.nc' component.

This method, in collaboration with a platform-unique 'hardware.h' file, connects the pins of the chips it is using to the appropriate outputs and inputs.

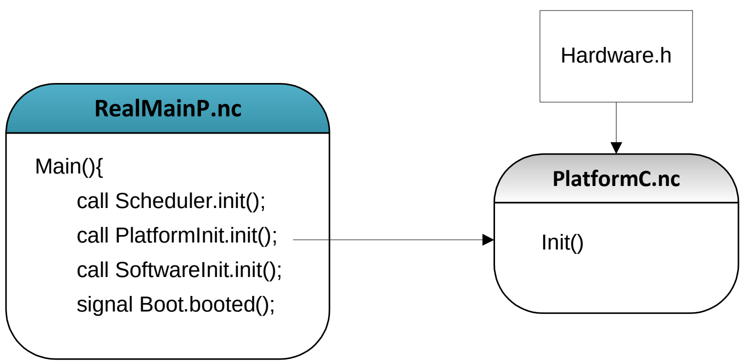


Figure 3.3 *TinyOS platform based boot sequence*

For the accelerometer board supplied with the BSNv3 node, a 'BSNv3DevKitsb' in defined in the TinyOS 'sensorboard' directory, which supplies the interface for accessing the accelerometer's readings. The board is connected to the mote in it's make target file, 'bsnv3.target' by the assignment: "SENSORBOARD ?= BSNv3DevKitsb".

We will add our new sensor board to the configuration or merge it with the existing accelerometer sensor board.

3.2.2 Driver for Gyroscope

For each sensor board that is used in the platform, a configuration and module .nc files are written for the various sensors the board contains, and a general module implementation, that ties all of them together.

In our case, since we need to add two gyroscopes, an XY gyro and an XZ gyro, we will have to create two modules:

GyroXYC.nc, GyroXYM.nc

GyroXZC.nc, GyroXZM.nc

Each configuration file (*C.nc) provides two 'Read<uint16_t>' interfaces, one for each axis. This interface consists of two methods:

1. command error_t read();

   (Basically a true/false specifying if new data is available)

2. event void readDone(value);

   (Copies the data received to the specified 'value' variable)

These interfaces are wired to a built in module written for receiving data from the ADC's, called 'AdcReadClientC()'. Note that the actual method for reading from the ADC already exists, and we will wire it to our new components.

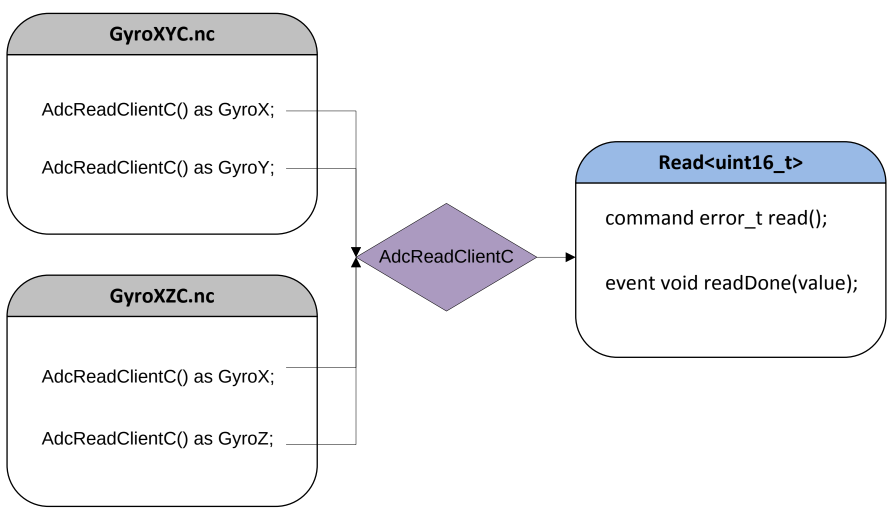


Figure 3.4 *new gyro modules with wired interface*

Each *M.nc file contains a configuration for each axis, that define the input channels for the sensor, in addition to other element necessary for reading. These are wired to the ADC client's configuration Interface, called 'AdcConfigure', by each gyro's configuration module.

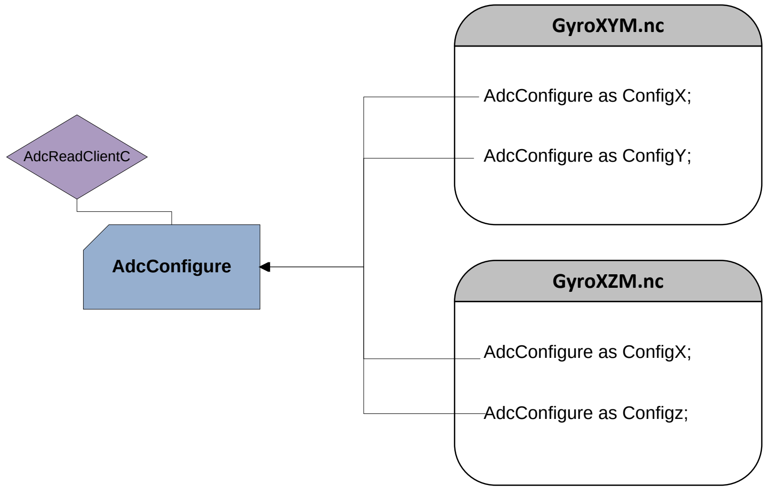


Figure 3.5 *ADC configuration for each axis in each gyro*

Another module, BSNv33DGyro(C/M).nc, will combine the ReadX, ReadY and ReadZ from the two other modules and will supply a full 3D gyro interface, which we can then use for our programs. It will also implement 'StdControl' start and stop methods, which regulate the power for the whole sensor board.

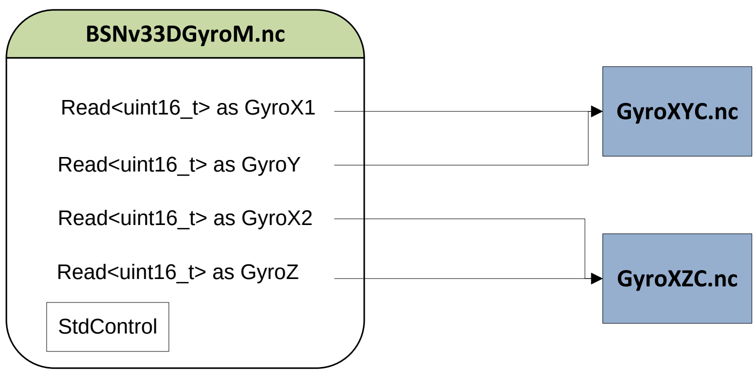


Figure 3.6 ***An overall module that supplies the full 3D interface***

3.2.3 Conversion & Filtering

3.2.3.1 Calibration

Since the accelerometer's output is influenced by the supplied voltage and battery power, there is a need to calibrate the accelerometer prior to use in order to get more accurate results.
For this purpose, we wrote an automatic calibration program using Matlab. For each axis, the program measures the accelerometer's output in zero and g gravity, and calculates the average over a given amount of output samples. This is done in order to compensate for the unstable measurements given by the accelerometer.

By assuming that the acceleration is linear, we can use a linear equation system to extract the actual acceleration from the output given by the accelerometer.

Using the averaged measurement results in each gravitational situation, we demand that:

$$\begin{aligned} a_X \cdot average(X_{acc}^0) + b_X &= 0 \\ a_X \cdot average(X_{acc}^g) + b_X &= 9.8 \\ a_Y \cdot average(Y_{acc}^0) + b_Y &= 0 \\ a_Y \cdot average(Y_{acc}^g) + b_Y &= 9.8 \end{aligned} \qquad (3.1)$$

X_{acc}^0, Y_{acc}^0 - the accelerometer's output samples in zero gravity.

X_{acc}^g, Y_{acc}^g - the accelerometer's output samples in g gravity.

The program calculates the coefficients for each one of the axes, and saves them. We can later use these coefficients to them to determine the actual acceleration by calculating:

$$\begin{aligned} Acc_X &= a_X \cdot X + b_X \\ Acc_Y &= a_Y \cdot Y + b_Y \end{aligned} \qquad (3.2)$$

3.2.3.2 Filters and Corrections

The accelerometer’s unstable measurements forced us to implement an adaptable Gaussian filter, in order to get smoother results.
The adaptable filter is composed of 6 Gaussian filters, scaling from the size of 5 to the size of 15 elements, where every filter is a weighted average of the current sample, FILTER_SIZE/2 samples before it and FILTER_SIZE/2 samples after it, and more weight is given to the elements closer to the current sample.
The filter size grows when we see consistency and shrinks back to 5 when we detect a drastic change in the acceleration.

As we reviewed the accelerometer’s output, we noticed that there are sometimes spikes in the results, that make it look as though the object is accelerating more than it actually does. This phenomenon happens once in a few samples, so we can pinpoint when it happens by comparing the sample’s value with its neighbors’ average.

When the distance between the two compared values exceeds 0.2 and the distance between the current sample's value and its previous' value exceeds what we conceive as a normal acceleration pace, we consider the value in question a spike, ignore it, and replace it with its neighbors' average. If we consider the current sample's successor a spike as well, we will replace it with the value of its previous.

$$
\begin{aligned}
&if \left|\frac{X_{succ} + X_{prev}}{2} - X_{curr}\right| > 0.2 \;\&\&\; \left|X_{prev} - X_{curr}\right| > 0.3 \; do \\
&\quad if \; |X_{succ} - X_{curr}| < 0.3 \; do \\
&\qquad X_{curr} = X_{prev} \\
&\quad else \; do \\
&\qquad X_{curr} = \frac{X_{prev} + X_{succ}}{2}
\end{aligned}
$$

Figure 3.7 *The pseudo code for the spike removal filter (where X is the samples buffer)*

Due to the fact that our accelerometer uses the earth's gravitational force to identify acceleration, when our module is not held directly parallel to the X and Y axes, the gravity affects the accelerometer's measurements, and we receive a false indication of acceleration.
In order to negate the gravity's affect, we need to keep track of the current angle of our moving body in accordance to the X and Y axes. When we receive an acceleration reading, we check our current angle and using simple trigonometry, calculate the exact offset of acceleration caused by gravity. We then subtract this offset from our reading in order to receive the real acceleration of the body.

3.2.4 Algorithm implementation

We implemented our algorithm using the Matlab software. The algorithm envelopes all the steps, starting with receiving the measurements, and ending with the display of the route.

3.2.4.1 Receiving the data

Receiving the raw data, calculating the measurements and storing them in appropriate buffers.

3.2.4.2 Filtering the measurements

The accelerometer's measurements are filtered for spikes and smoothed with the Gaussian Filter, as detailed in the filtering section (3.2.4.2).
The results of the gyroscope are also filtered using a basic averaging filter.

3.2.4.3 Converting the angular velocity measured by the gyroscope

The gyroscope measures the angular velocity in each direction, during the sample interval. In order to obtain the new angles, we use the following equation, for each coordinate:

$$\theta = \int_{t_0}^{t} \dot{\theta}(\tau) d\tau \qquad (3.3)$$

3.2.4.4 Converting to earth coordinates

Implementing matrix and calculation as explained in section 2.3.2, that results in [X,Y,Z] coordinates in relation to earth axes.

3.2.4.5 Compensating for gravity

Extracting the current angle of the object in relation to earth's X and Y axes, computing the affects of the gravity on the accelerometer's measurement, and reducing it from the acceleration we calculated previously.

3.2.4.6 Calculating the progress vector

Calculating the distance the object traveled in that time interval, using the accelerometer's measurements, calculating the direction of progress using the angles previously calculated, and combining the two to extract the progress vector.

3.2.4.7 Displaying the results

3.2.5 GUI

We use the Matlab application to run our algorithm and display the results graphically using its 'Figure' tools. After analyzing the data and calculating the object's current position, we'll display the course the object makes in real-time.

4. Results and Conclusions

4.1 Calibration Testing

As showed in section 3.2.3.1, we calibrated the accelerometer's data before we started the test its measurements.
The results we received at first, before the calibration was done, were inaccurate. The inaccuracy was

caused by the measurements' drift, resulted from internal heating and the accelerometer's mechanism, and also by inconsistent readings, even when the module was stationary.

These circumstances brought us to choose to calibrate the accelerometer, after which the accuracy of our results improved immensely.

	Min	Max	Avg.	Delta
X				
Y				

Table 4.1 *results of calibration*

4.2 Accelerometer's Filter Testing

We looked at the accelerometer's readings for some random movement, and displayed the results before and after the filtering process. We did this to test if the filter works correctly.

At first we applied only the Gaussian filter on the accelerometer's results. We noticed that sometimes, a measurement spike, that falsely indicated a certain acceleration value, caused the results to chance, although the filter was applied. This alerted us to the fact that we need remove the spikes by altering the readings from the received measurement to an average of its surrounding samples.
After we applied this alteration, our results were much smoother, but we noticed that sometimes spikes arrive one directly after another, causing a distortion of the filtered result. We changed the spike filter to also address this problem, and this we were content with the results, as can be seen in the figure below:

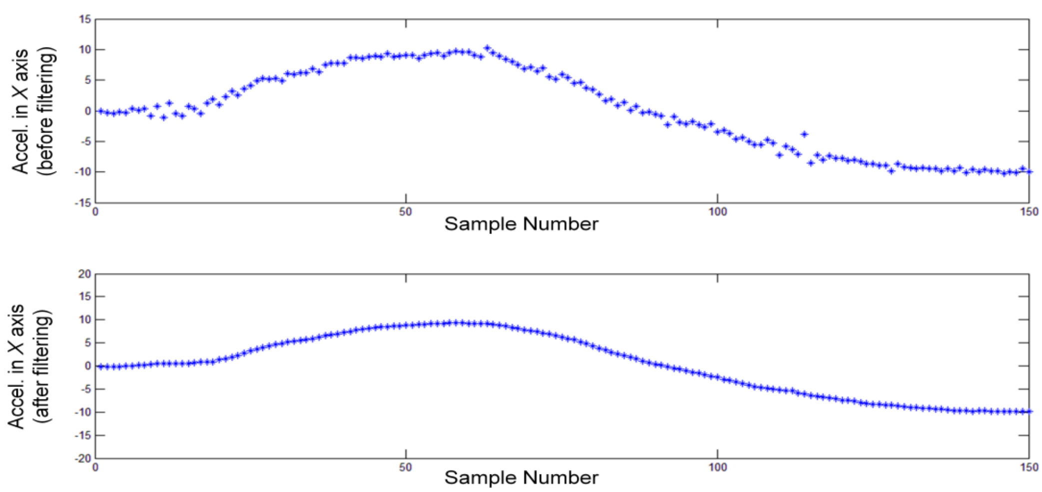


Figure 4.0.1 *Accelerometer's X axes results before and after filtering*

As the graphs show, after we apply the filter the results are smoother and less noisy, so we can proceed to the next level.

4.3 Gyro Hardware Testing

According to the gyroscope specifications sheet, we measured, using an oscilloscope, the input and output voltage of the gyroscope, to make sure we receive correct readings.

The input voltage the gyroscope expects in order to function correctly is between 2.7V and 3.6V. When we tested the gyroscope's input voltage, using the oscilloscope, we noticed that the gyroscope's supply voltage is only 2.2V. We concluded that the battery given to us with the BSN Kit doesn't supply the required voltage to activate both gyroscope and accelerometer properly. Thus, we decided to replace the battery holder with one that holds 2 standard AA 1.5V batteries. After doing so we got a supplied voltage of 3V for the gyroscope, which as mentioned before is enough.

When testing the output voltage of each of the gyroscope's axes, we noticed a lot of noise, when using the oscilloscope. We then decided to replace the original 2.2nF capacitors we used in the gyroscope's supporting board with 1uF capacitors, thus getting better results.

4.4 Route Tracking

In order to test that our algorithm, using the readings from the combined gyroscope and accelerometer module, is able to detect an actual route in 3D space, we mounted the module on a remote controlled toy car, which we then moved in various routes, and checked whether the software correctly displayed the car's path.

Our first test was to move the car in a continuous circular movement, and check how well the route is displayed.

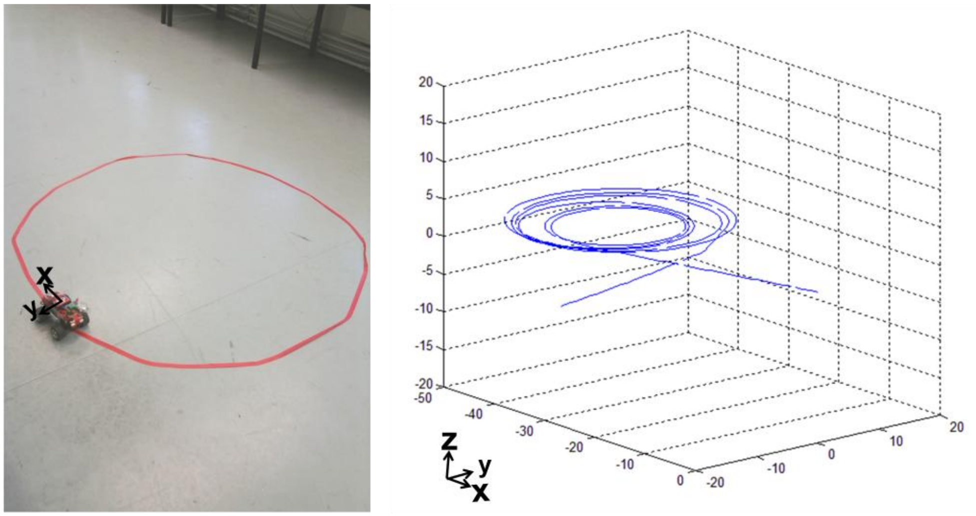


Figure 4.0.2 *A circular route and its display via Matlab*

As we can see from the Matlab plot our algorithm successfully managed to identify that the car moved in a circular route.

We proceeded to moving the car in different routes, each time checking if the correct path was displayed, and making small adjustments to the filters accordingly. By doing so we were able to achieve satisfying results.

The following figure contains screenshots from one of these videoed tests, in which we moved the car in a random path, and checked the progress on the display.

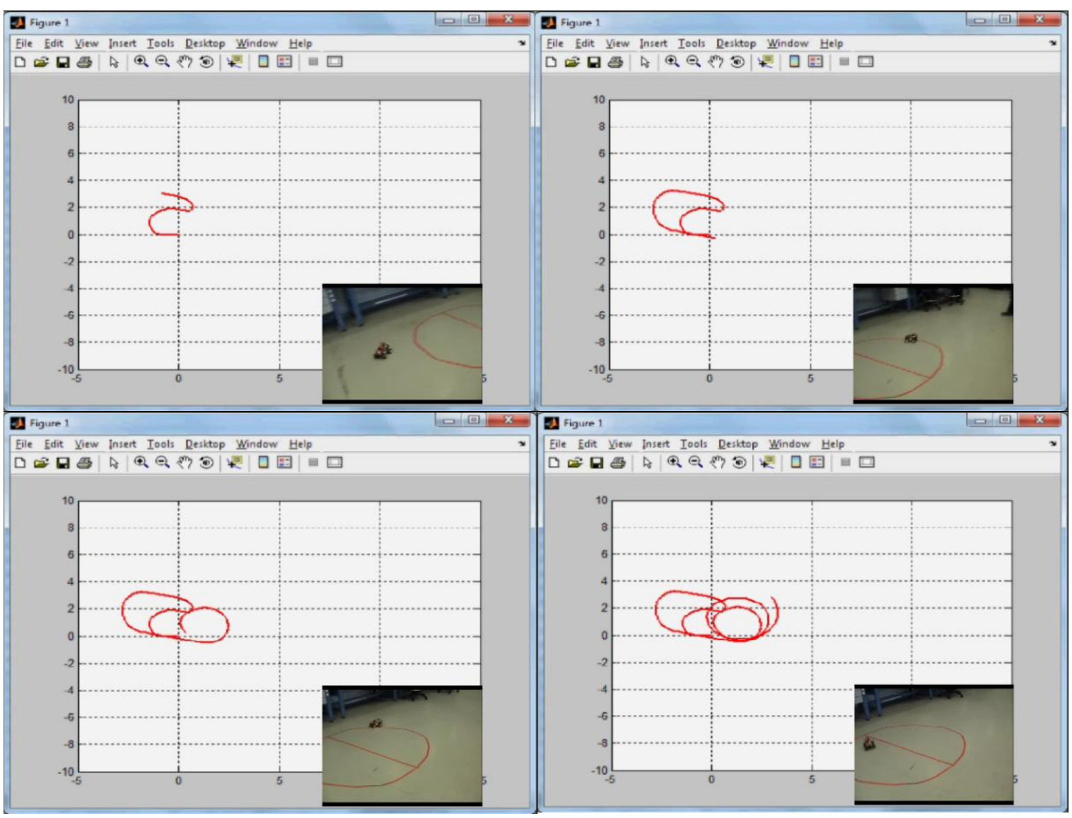


Figure 4.0.3 *Screenshots from the video depicting the progress in Matleb's display, as the car makes its route*

Our tests were performed in a 2D space, using only one of the gyroscopes (Pitch-Yaw), but our module and software are designed to work fully well in 3D space.