



STMicroelectronics SensorTile Tutorial:

Real Time Inertial Sensing with SensorTile



Table of Contents

1. INTRODUCTION TO THIS TUTORIAL	3
LIST OF REQUIRED EQUIPMENT AND MATERIALS.....	3
PREREQUISITE TUTORIALS.....	3
2. THE DISPLACEMENT MEASUREMENT SYSTEM	4
3. SYSTEM SAMPLING FREQUENCY AND SENSOR HANDLER CONFIGURATION	8
4. DIGITAL SIGNAL PROCESSING DISCRETE TIME FILTERS	11
5. DISPLACEMENT COMPUTATION	13
6. DISPLACEMENT REAL TIME DISPLAY	19
7. REAL TIME INERTIAL SENSING PROJECT INSTALLATION	21
SENSORTile EMBEDDED ML SYSTEM INSTALLATION.....	21
8. REAL TIME INERTIAL SENSING VISUALIZATION: MAC OSX	24
9. REAL TIME INERTIAL SENSING VISUALIZATION: WINDOWS	27
10. REAL TIME INERTIAL SENSING VISUALIZATION: DEMONSTRATION OF DISPLACEMENT SENSING ...	31
11. GRAVITATIONAL ACCELERATION AND DISPLACEMENT MEASUREMENT	35
12. ASSIGNMENT: MOTION SENSING FILTER DESIGN	37



1. Introduction to This Tutorial

Previous Tutorials have introduced inertial motion sensing, finite state machine-based posture recognition, and digital signal processing. In this tutorial, we will develop capability for computing motion displacement with accelerometer systems. This will also include a real time visualization system of displacement in the X- and Y-Axis directions.

This Tutorial is intended to provide both experience in development of inertial sensing applications as well as to provide intuition for the contributions to inertial sensing including the those of motion displacement as well as gravitational acceleration.

For more information regarding the SensorTile board, please open the following link.

www.st.com/sensortile

List of Required Equipment and Materials

- 1) 1x STMicroelectronics SensorTile kit.
- 2) 1x STMicroelectronics Nucleo Board.
- 3) 1x Personal Computer with two USB type-A inputs OR you must have a powered USB hub.
- 4) 1x USB 2.0 A-Male to Micro-B Cable (micro USB cable).
- 5) 1x USB 2.0 A-Male to Mini-B Cable (mini USB cable).
- 6) Network access to the Internet.
- 7) 1x Small piece of foam pad

Prerequisite Tutorials

It is recommended that users have completed and are familiar with the contents of the following tutorials before proceeding.

- 1) Tutorials including 1, 2, and 3.

Your instructor will ensure you have the required background regarding motion sensing, digital signal processing, and basic physics of acceleration, velocity, and displacement.



2. The Displacement Measurement System

The detection and computation of displacement is determined by the integration of acceleration. The SensorTile system includes three axes of acceleration measurements, X, Y, and Z as shown in Figure 1.

If the Z-axis is oriented in the vertical direction, and aligned with the gravitational acceleration vector, then the projection of the gravitational acceleration vector on the X and Y axes will be zero. X- and Y-axis acceleration will indicate zero value if the SensorTile device is either stationary or moving at a constant velocity. However, if an angle appears between the Z-axis and the gravitational acceleration vector, then a large acceleration signal appears in either or both of X- and Y-axes. This will indicate velocity and displacement *whether the device is stationary or is in motion*.

This Tutorial is directed to measurement of SensorTile motion in a horizontal plane.

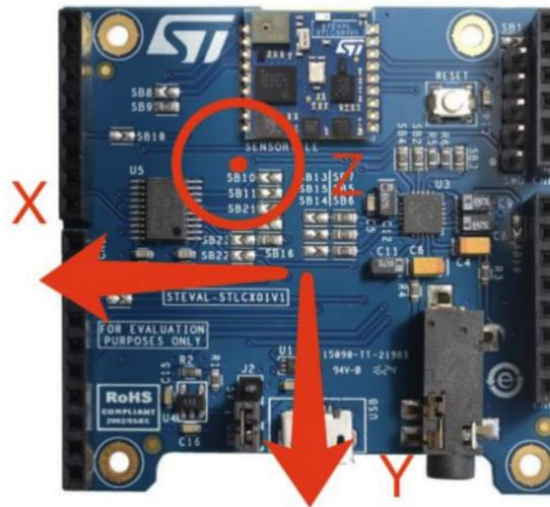


Figure 1. The SensorTile acceleration axis orientations



Considering acceleration $a_X(t)$ in the X-axis direction, measured velocity change in X-axis, $v_X(t)$ is

$$v_X(t) = v_X(t = 0) + \int_{\tau=0}^{\tau=t} a_X(\tau) d\tau$$

Now, since the SensorTile digital system samples at discrete time intervals, determined by the sampling period, T_{Sample} , the angle is determined by a sum, replacing the continuous time integral above. Also, time, t , now becomes nT_{Sample} , where n is the index value over all samples. Then, a_X is sampled also at times nT_{Sample} .

Now, computation of this definite integral over discrete values is accomplished by the trapezoidal rule. As an example, consider the function displayed in Figure 2. The continuous function, $R(t)$, is sampled at the time points $t = a$ and $t = b$. An estimate for the definite integral between these points (the area under the function between the sample points) is

$$\int_{\tau=a}^{\tau=b} a_X(\tau) d\tau = (t_1 - t_2) \left(\frac{a_X(t_1) + a_X(t_2)}{2} \right)$$

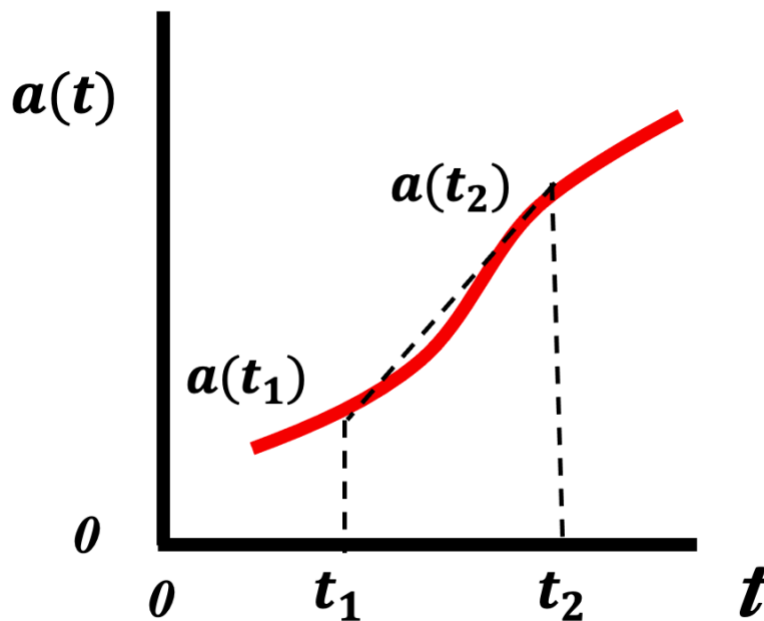


Figure 3. Example of function $f(x)$ with the trapezoidal approximation estimating the area under the function between the interval between $x = a$ and $x = b$.



Now, the difference, $(t_1 - t_2)$, is simply the sampling period, T_{Sample}

Thus, the velocity computation for each axis is

$$v_x(n) = v_x(n = 0) + \sum_{i=1}^{i=n} \frac{(a_x(i - 1) + a_x(i))}{2} T_{Sample}$$

Similarly,

$$v_y(n) = v_y(n = 0) + \sum_{i=1}^{i=n} \frac{(a_y(i - 1) + a_y(i))}{2} T_{Sample}$$

and

$$v_z(n) = v_z(n = 0) + \sum_{i=1}^{i=n} \frac{(a_z(i - 1) + a_z(i))}{2} T_{Sample}$$

Displacement is determined by performing a time integral of velocity.

$$d_x(t) = d_x(t = 0) + \int_{\tau=0}^{\tau=t} v_x(\tau) d\tau$$

Since the SensorTile acceleration data source is a discrete time, sampled signal, where acceleration is sampled at a period of T_{Sample} and at a frequency of $1/T_{Sample}$, a discrete trapezoidal integration is computed on velocity. Displacements over each of the three axes are:

$$\begin{aligned} d_x(n) &= d_x(n = 0) + \sum_{i=1}^{i=n} \frac{(v_x(i - 1) + v_x(i))}{2} T_{Sample} \\ d_y(n) &= d_y(n = 0) + \sum_{i=1}^{i=n} \frac{(v_y(i - 1) + v_y(i))}{2} T_{Sample} \\ d_z(n) &= d_z(n = 0) + \sum_{i=1}^{i=n} \frac{(v_z(i - 1) + v_z(i))}{2} T_{Sample} \end{aligned}$$



3. Compensation for Sensor Offset

The accelerometer sensor hosted on the SensorTile system is a state-of-the-art device with accuracy equal to the best available devices of its type.

This device is accurately calibrated to measure acceleration. The sensor gain selected provides a measurement in units of milli-g where $g = 9.81 \text{ m/sec}^2$. The sampling rate of this project application is set at 1000 samples per second corresponding to a sampling period, $T_{\text{sample}} = 0.001$ seconds.

Now, all motion sensor systems display a finite error, referred to as offset. Specifically, if the sensor is exposed to zero acceleration input in one of its axes, its acceleration measurement output is nonzero. This nonzero value leads to computational errors since it is integrated over time thus indicates presence of a velocity and after integration of velocity, a displacement .

The offset error signal may be removed directly. This is accomplished by measuring the offset signal for each sensor axis during a time when the sensor is motionless. Then, this offset signal is subtracted from all subsequent measurements. Some applications do not offer a knowledge of a time interval when the sensor system is motionless presenting an additional challenge.



4. System Sampling Frequency and Sensor Handler Configuration

The Datalog application operates with a sensor signal sampling period of 1 milli-seconds, or 1000 Hz.

```
/* Data acquisition period [ms] */
#define DATA_PERIOD_MS (1)           // Acceleration measurement period in milliseconds

#define HIGH_PASS_FILTER_FREQUENCY 0.2 // Displacement computation high pass corner // frequency
#define LOW_PASS_FILTER_FREQUENCY 1    // Displacement computation low pass corner // frequency

#define DISPLAY_INTERVAL 10000        // Display period in milliseconds
```

In addition, this application enables only the **Accelero_Sensor_Handler()** in **main.c** as shown in. You will be able to enable rate gyroscope signals in the next Tutorial.

```
int main( void )
{
    uint32_t msTick, msTickPrev = 0;

    float acc_x, acc_y;
    float acc_x_prev, acc_y_prev;
    float acc_x_filter, acc_y_filter;
    float acc_x_filter_prev, acc_y_filter_prev;
    float disp_x = 0;
    float vel_x = 0;
    float vel_x_prev = 0;
    float vel_x_filter_prev = 0;
    float vel_x_filter = 0;
```




```

float disp_x_prev = 0;
float disp_x_filter_prev = 0;
float disp_x_filter = 0;
float disp_y = 0;
float vel_y = 0;
float vel_y_prev = 0;
float vel_y_filter_prev = 0;
float vel_y_filter = 0;
float disp_y_prev = 0;
float disp_y_filter_prev = 0;
float disp_y_filter = 0;
int accel_offset_x = 0;
int accel_offset_y = 0;
int i, j, initialize_state;
float sample_period;
char dataOut[256];
float W_HP, I_HP, iir_HP_0, iir_HP_1, iir_HP_2;
float W_LP, I_LP, iir_LP_0, iir_LP_1, iir_LP_2;
uint8_t id;
uint8_t status;
SensorAxes_t acceleration;

BSP_ACCELERO_Get_Instance( LSM6DSM_X_0_handle, &id );
BSP_ACCELERO_IsInitialized( LSM6DSM_X_0_handle, &status );
if (status == COMPONENT_ERROR){
    sprintf( dataOut, "Sensor Initialization Error");
}

```



```
CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));  
}
```



5. Digital Signal Processing Discrete Time Filters

```

/* Indicate that displacement variables not initialized */
initialize_state = 0;

/*
 * Define sample period
 */
sample_period = (float)(DATA_PERIOD_MS)/1000;

/*
 * Compute high pass filter coefficients
 */

W_HP = 2 * 3.1416 * HIGH_PASS_FILTER_FREQUENCY;
I_HP = 2/(W_HP * sample_period);
iir_HP_0 = 1 - (1/(W_HP + I_HP));
iir_HP_1 = -iir_HP_0;
iir_HP_2 = (1/(1 + I_HP))*(1 - I_HP);

/*
 * Compute low pass filter coefficients
 */

```



```
W_LP = 2 * 3.1416 * LOW_PASS_FILTER_FREQUENCY;  
I_LP = 2/(W_LP * sample_period);  
iir_LP_0 = 1 - (1/(W_LP + I_LP));  
iir_LP_1 = iir_LP_0;  
iir_LP_2 = (1/(1 + I_LP))*(1 - I_LP);
```



6. Displacement Computation

```
HAL_Init();

/* Configure the system clock */
SystemClock_Config();

/* Initialize RTC */
RTC_Config();
RTC_TimeStampConfig();

/* enable USB power on Pwrctrl CR2 register */
HAL_PWREx_EnableVddUSB();

if(SendOverUSB) /* Configure the USB */
{
    /*** USB CDC Configuration ***/
    /* Init Device Library */
    USB_D_Init(&USB_D_Device, &VCP_Desc, 0);
    /* Add Supported Class */
    USB_D_RegisterClass(&USB_D_Device, USB_D_CDC_CLASS);
    /* Add Interface callbacks for AUDIO and CDC Class */
    USB_D_CDC_RegisterInterface(&USB_D_Device, &USB_D_CDC_fops);
    /* Start Device Process */
    USB_D_Start(&USB_D_Device);
}
```



```
/* Configure and disable all the Chip Select pins */
```

```
Sensor_IO_SPI_CS_Init_All();
```

```
/* Initialize and Enable the available sensors */
```

```
initializeAllSensors();
```

```
enableAllSensors();
```

```
while (1)
```

```
{
```

```
/* Get sysTick value and check if it's time to execute the task */
```

```
msTick = HAL_GetTick();
```

```
if(msTick % DATA_PERIOD_MS == 0 && msTickPrev != msTick)
```

```
{
```

```
msTickPrev = msTick;
```

```
if (SendOverUSB) {
```

```
/* Initialize LED */
```

```
BSP_LED_Init(LED1);
```

```
}
```

```
if((msTick % (DATA_PERIOD_MS*10000)) == 0 || initialize_state == 0){
```



```

/*
 * This sequence will execute at the start of execution since
 * initialize_state is set to zero. After the first execution,
 * initialize_state is set to 1 and execution occurs only when
 * the timer tick value is a multiple of DISPLAY_INTERVAL
 */

initialize_state = 1;

/*
 * Provide notification of reset of displacement offset
 */

for (j = 0; j < 5; j++){
    BSP_LED_Off(LED1);
    for (i = 0; i < 80; i++){
        sprintf( dataOut, "_");
        CDC_Fill_Buffer(( uint8_t *)dataOut, strlen( dataOut ));
    }
    sprintf( dataOut, "\r\n");
    CDC_Fill_Buffer(( uint8_t *)dataOut, strlen( dataOut ));
    HAL_Delay(500);
}

/*
 * Initialize displacement and filter values
 */

acc_x = 0;
acc_y = 0;

```



```

    acc_x_prev = 0;
    acc_y_prev = 0;
    acc_x_filter_prev = 0;
    acc_y_filter_prev = 0;

    disp_x = 0;
    vel_x = 0;
    vel_x_prev = 0;
    vel_x_filter_prev = 0;
    disp_x_prev = 0;
    disp_x_filter_prev = 0;
    disp_y = 0;
    vel_y = 0;
    vel_y_prev = 0;
    vel_y_filter_prev = 0;
    disp_y_prev = 0;
    disp_y_filter_prev = 0;

    /*
     * Measure offset values
     */

    BSP_ACCELERO_Get_Axes( LSM6DSM_X_0_handle, &acceleration );

    accel_offset_x = acceleration.AXIS_X;
    accel_offset_y = acceleration.AXIS_Y;
}

if (initialize_state == 1){
    BSP_LED_On(LED1);
}

```




```
BSP_ACCELERO_Get_Axes( LSM6DSM_X_0_handle, &acceleration );
```

```
/*
```

```
 * Compute displacement and velocity by integration
```

```
*/
```

```
acc_x = acceleration.AXIS_X - accel_offset_x;
```

```
acc_y = acceleration.AXIS_Y - accel_offset_y;
```

```
acc_x_filter = iir_LP_0 * acc_x + iir_LP_1 * acc_x_prev + iir_LP_2 *  
    acc_x_filter_prev;
```

```
acc_y_filter = iir_LP_0 * acc_y + iir_LP_1 * acc_y_prev + iir_LP_2 *  
    acc_y_filter_prev;
```

```
vel_x = vel_x + (acc_x_filter + acc_x_filter_prev) * sample_period/2;
```

```
vel_y = vel_y + (acc_y_filter + acc_y_filter_prev) * sample_period/2;
```

```
acc_x_prev = acc_x;
```

```
acc_x_filter_prev = acc_x_filter;
```

```
acc_y_prev = acc_y;
```

```
acc_y_filter_prev = acc_y_filter;
```

```
/*
```

```
 * Apply high pass filter to velocity
```

```
*/
```

```
vel_x_filter = vel_x * iir_HP_0 + vel_x_prev * iir_HP_1 -  
    vel_x_filter_prev*iir_HP_2;
```

```
vel_y_filter = vel_y * iir_HP_0 + vel_y_prev * iir_HP_1 -  
    vel_y_filter_prev*iir_HP_2;
```



```

/* Integrate velocity to compute displacement */

disp_x = disp_x + (vel_x_filter + vel_x_filter_prev) * sample_period/2;
disp_y = disp_y + (vel_y_filter + vel_y_filter_prev) * sample_period/2;

vel_x_filter_prev = vel_x_filter;
vel_x_prev = vel_x;

vel_y_filter_prev = vel_y_filter;
vel_y_prev = vel_y;

/*
 * Apply high pass filter to displacement
 */

disp_x_filter = disp_x*iir_HP_0 + disp_x_prev * iir_HP_1 -
    disp_x_filter_prev*iir_HP_2;
disp_x_filter_prev = disp_x_filter;
disp_x_prev = disp_x;

disp_y_filter = disp_y*iir_HP_0 + disp_y_prev * iir_HP_1 -
    disp_y_filter_prev*iir_HP_2;
disp_y_filter_prev = disp_y_filter;
disp_y_prev = disp_y;

/*
 * Compute displacement index for plotting
 * Accelerometer sensor output values are in units of milli-g.
 * Acceleration value is, therefore, 9.81 mm/s/s = 0.981 cm/s/s
 * Thus, computed displacement is scaled by this value
 */

plot_dual_time_series((int)((1/0.981) * disp_x_filter),
    ((int)(1/0.981) * disp_y_filter));

```



```
}
```

7. Displacement Real Time Display

Real Time display of displacement data is rendered with a terminal application. This includes the Mac OSX **screen** utility or the Windows **putty** terminal application.

The `plot_dual_time_series()` function provides this capability. While this animation is limited in some respects, this approach offers simplicity and avoids installation of animation applications on various platforms.

```
void plot_dual_time_series(int plot_1_disp, int plot_2_disp)
{
    int i;
    int plot_1_tab, plot_1_add, plot_2_tab, plot_2_add;
    plot_1_disp = 40 + plot_1_disp;
    plot_2_disp = 40 + plot_2_disp;
    char dataOut[120];
    /* Limit displacement to 80 character terminal display */
    if (plot_1_disp < 0){
        plot_1_disp = 0;
    }
    if (plot_1_disp > 80){
        plot_1_disp = 80;
    }
    if (plot_2_disp < 0){
        plot_2_disp = 0;
    }
    if (plot_2_disp > 80){
        plot_2_disp = 80;
    }
    /* Determine number of 8-space tab characters equal plot position */
    plot_1_tab = floor(plot_1_disp/8);
    plot_2_tab = floor(plot_2_disp/8);
    /* Determine remaining character spaces required */
```



```

    plot_1_add = plot_1_disp - plot_1_tab * 8;
    plot_2_add = plot_2_disp - plot_2_tab * 8;

    /* Print tabs */
    for (i = 0; i < plot_1_tab; i++){
        sprintf( dataOut, "\t");
        CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));
    }

    /* Print spaces */
    for (i = 0; i < plot_1_add; i++){
        sprintf( dataOut, " ");
        CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));
    }

    /* Print symbol */
    sprintf( dataOut, "X\r\n");
    CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));

    /* Print tabs */
    for (i = 0; i < plot_2_tab; i++){
        sprintf( dataOut, "\t");
        CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));
    }

    /* Print spaces */
    for (i = 0; i < plot_2_add; i++){
        sprintf( dataOut, " ");
        CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));
    }

    /* Print symbol */
    sprintf( dataOut, "Y\r\n");
    CDC_Fill_Buffer(( uint8_t * )dataOut, strlen( dataOut ));
}

```



8. Real Time Inertial Sensing Project Installation

As in other EmbeddedML projects, this project, is based on an extension of the DataLog application to include EmbeddedML.

SensorTile EmbeddedML System Installation

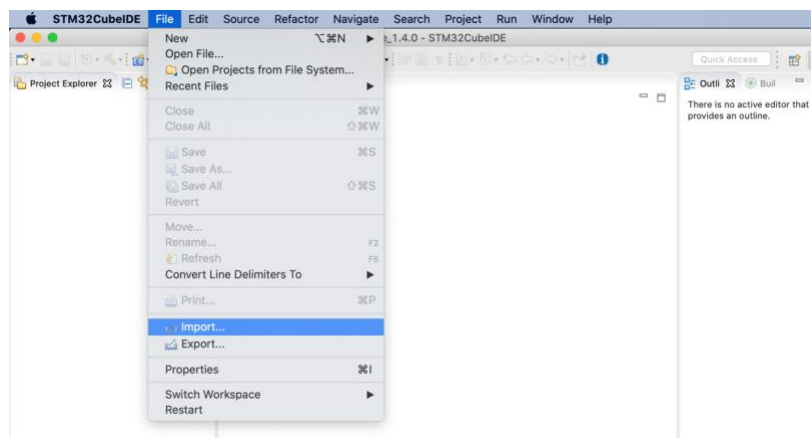
1. Download the project from Google Drive

https://drive.google.com/file/d/1dles72C_EWzZ8kti5D31D62zuT9FVR7g/view?usp=sharing

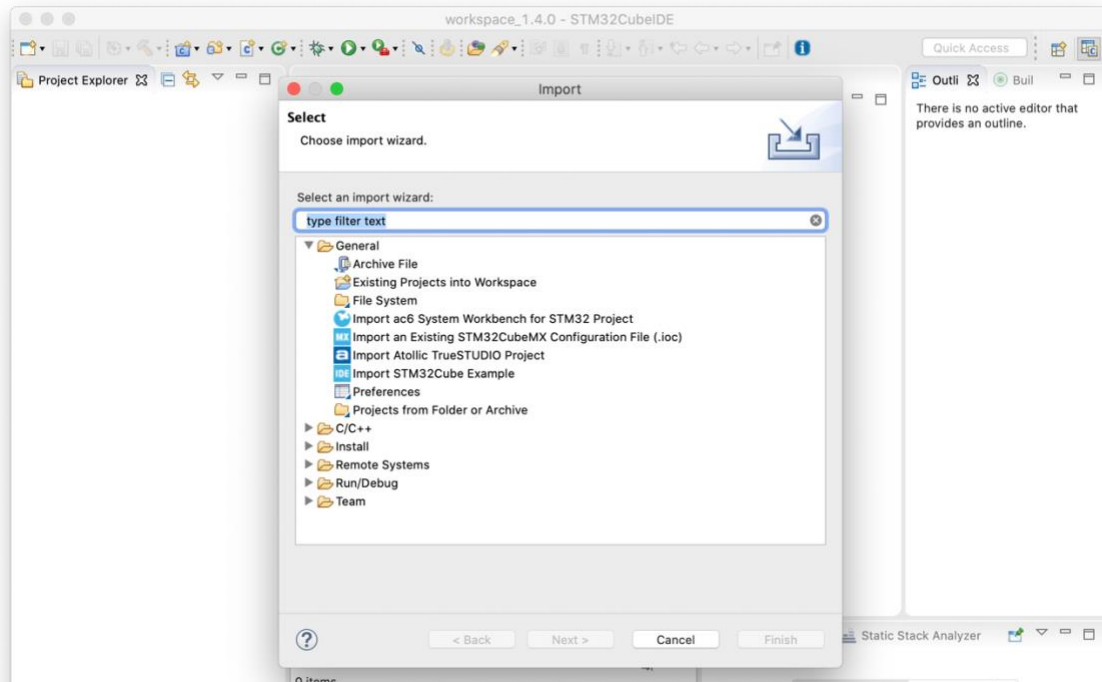
and decompress it to your working directory.

Note: Please download this project above. Please do not use the Projects from previous Tutorials.

2. Open the STM32CubeIDE, as instructed in the Tutorial 1. Select the same workspace as in Tutorial 1.
3. Once the IDE is open, first remove your current project.
4. Then, import the new Tutorial 10
5. Now, start the STM32CubeIDE Application. Select File > Import

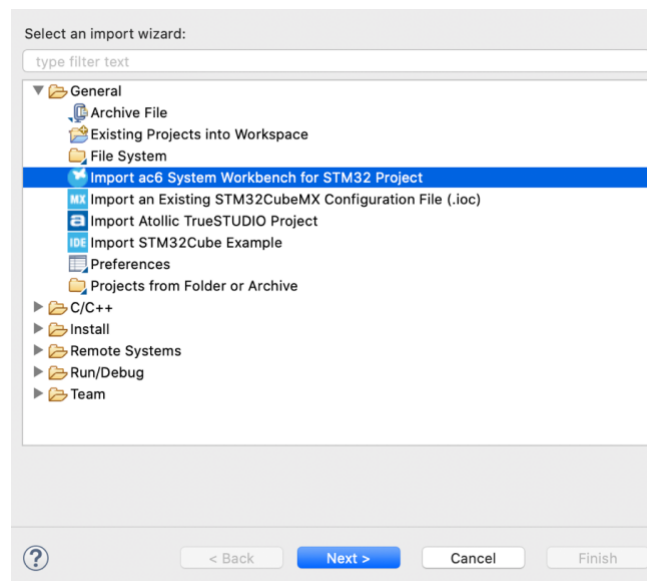


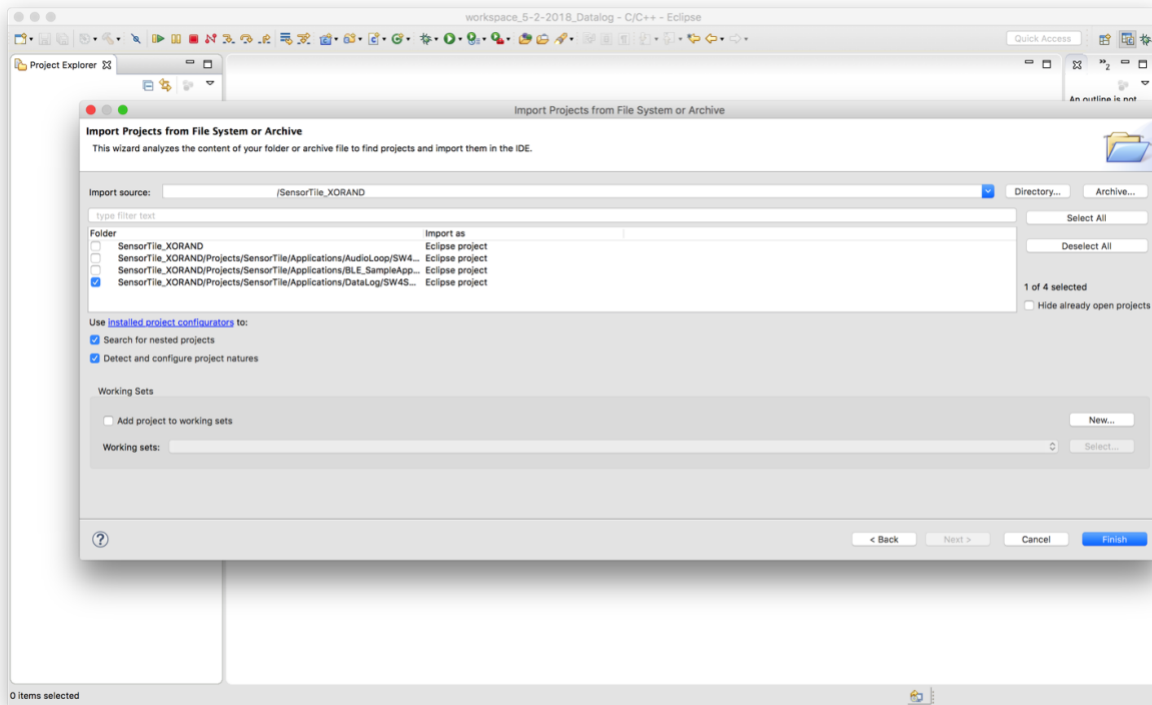
6. Then, Select General



7. Select Import ac6 System Workbench for STM32 Project

NOTE THIS SELECTION IS CRITICAL





8. Then, select Project > Clean
9. Then, select Project > Build All
10. Then select **Run > Debug As > STM32 Cortex-M C/C++**



9. Real Time Inertial Sensing Visualization: Mac OSX

1. To enable Real Time Visualization on Mac OSX, start a Terminal session



2. First, determine the Device File interface assigned to the SensorTile
3. Enter this command:
ls /dev/tty.usbmodem*
4. This screen will appear. Mac OSX assigns Device File numbers to each serial port.
5. The SensorTile Device File ends in numeral 1.





6. Now, the screen command can be used to observe data from the SensorTile.
7. There are two methods for entering this. One is by copy and paste of the address
 - a) Using the cursor, you may copy the device file address to the clipboard
 - b) Then, enter screen and paste the address as below



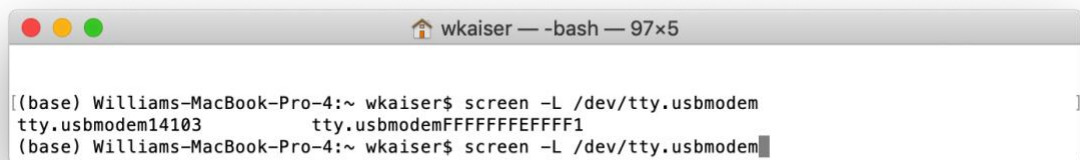
```
wkaiser — -bash — 97x11
(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodemFFFFFFFFFFFF1
```

8. The second method is to use the Tab-Autocomplete method
 - a) First, enter



```
wkaiser — -bash — 97x5
(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodem
```

- b) Press the tab key – this will show the options for Autocomplete



```
wkaiser — -bash — 97x5
[(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodem
tty.usbmodem14103      tty.usbmodemFFFFFFFFFFFF1
(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodem]
```



c) Type F

```
wkaiser — -bash — 97x5
[(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodem
tty.usbmodem14103          tty.usbmodemFFFFFFFFFFFF1
[(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodemF
```

d) Type the tab key, this will Autocomplete the address

```
wkaiser — -bash — 97x5
[(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodem
tty.usbmodem14103          tty.usbmodemFFFFFFFFFFFF1
[(base) Williams-MacBook-Pro-4:~ wkaiser$ screen -L /dev/tty.usbmodemFFFFFFFFFFFF1
```

9. Enter Return and data from the screen command will be visible

```

Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y
X
Y

```

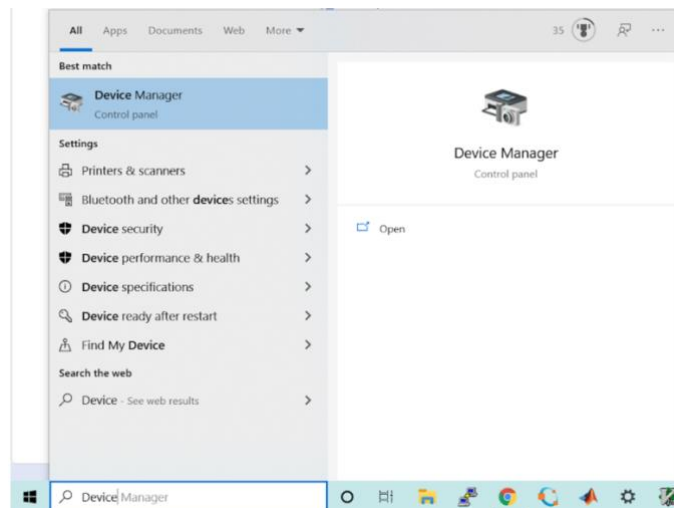
10. This session can be terminated with the exit command sequence.

11. Hold down the Ctrl key and then press the “a” key and then the “\” key

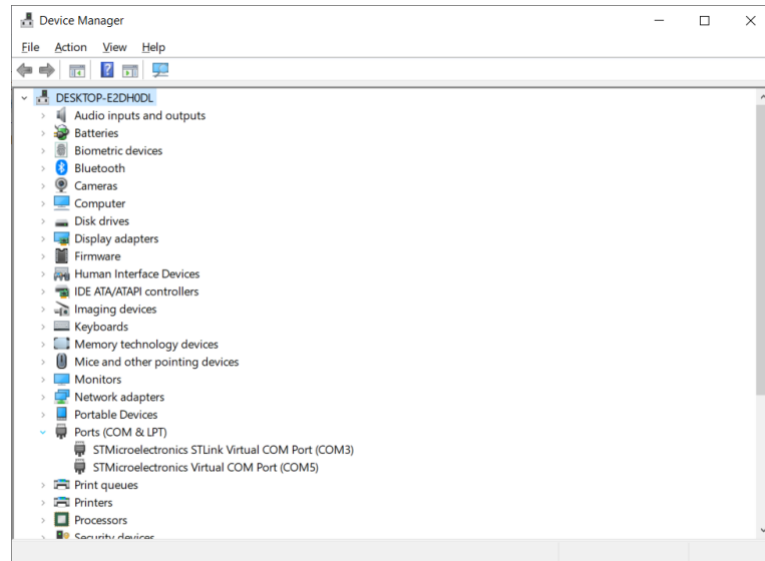


10. Real Time Inertial Sensing Visualization: Windows

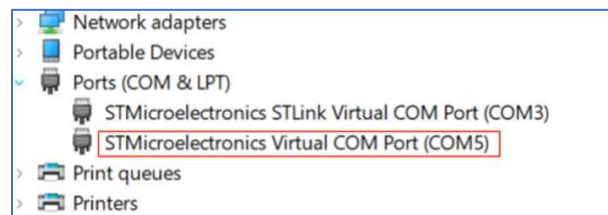
- 1) To enable a Real Time Visualization session on Windows, proceed with the Putty application as in Tutorial 1.
- 2) proceed to determine the Serial Port assigned to the SensorTile by the Windows operating system.
- 3) Open up the Device Manager by entering “device manager” in the Windows search bar.



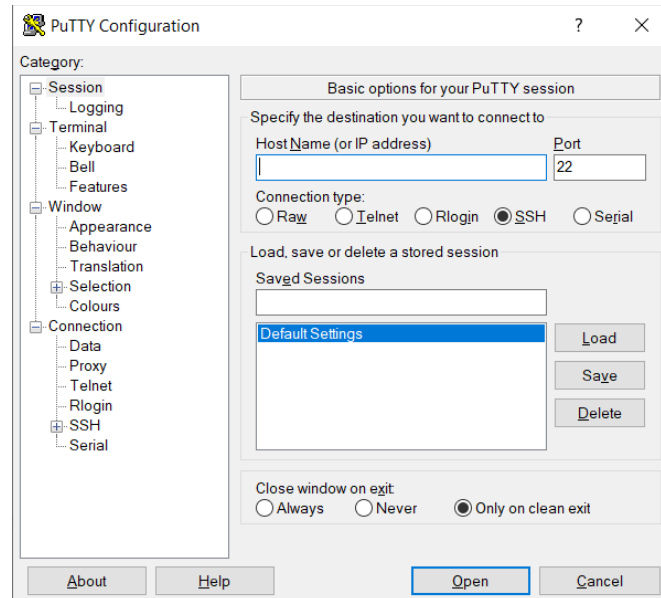
- 4) Expand the **Ports (COM and LPT)** selection



- 5) Find the COM port number indicated for the STMicroelectronics Virtual COM Port. In this example, the Port is **COM5**.

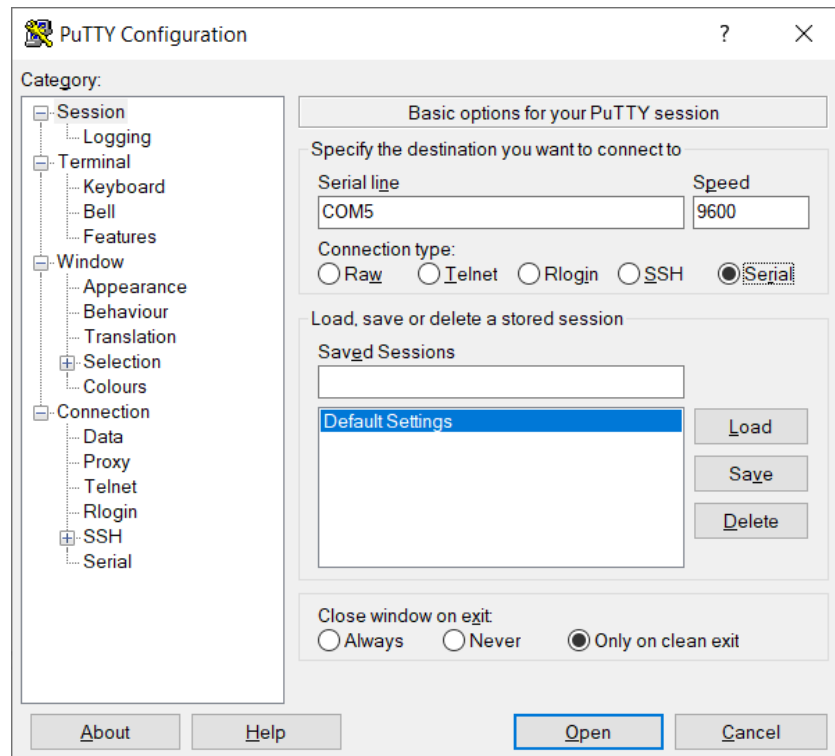


- 6) Start the Putty Application. This may be started by entering putty in the Windows Search Bar.
- 7) This screen will appear



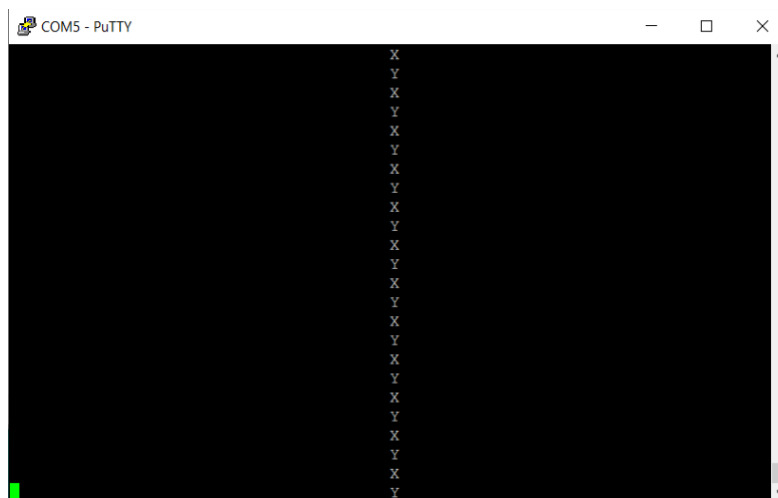


8) Check the **Serial** checkbox, enter the **COM Port** found above, and enter **9600** for Baud Rate.



9) Select Open

10) Data will now appear as shown below.





11. Real Time Inertial Sensing Visualization: Demonstration of Displacement Sensing

As presented in Section 2, acceleration may be measured and then integrated with respect to time to yield a measurement of velocity. Then, velocity may be integrated with respect to time to produce a measurement of displacement.

It is essential, however, to determine the origin of acceleration signals. Acceleration is produced by motion as well as by gravitation. Specifically, an accelerometer that is motionless may be expected to indicate zero acceleration only if its axes of measurement are not aligned with the gravitational acceleration vector, normal to the Earth's surface, and with a value of

$$g = 9.81 \text{ m/sec}^2$$

Consider an accelerometer sensor with an axis, labeled X, that is aligned perpendicular to the gravitational vector. Now, a motion of acceleration of 0.1 m/sec^2 executed for 1 second followed by a deceleration of 0.1 m/sec^2 executed for 1 second will result in a displacement of 10 cm with

$$d_x = \frac{1}{2} a_x t^2 + \frac{1}{2} a_x t^2 = 0.1 \text{ m}$$

This displacement and acceleration are typical of a human gesture, for example.

However, now consider an example where the sensor system is motionless and also where the X-axis is now oriented at an angle of 30 degrees relative to the horizontal plane. Then, the acceleration is

$$a_x = g \sin 30 = 4.9 \text{ m/sec}^2$$

Then, the apparent displacement in 2 seconds would be 19.6m.

This demonstrates a critical challenge for inertial sensing. Specifically, accurate measurement of the inclination of a sensor relative to the gravitational acceleration vector is required to achieve accurate estimation of displacement.

Now, view the video below to determine how to orient and move the SensorTile. In this example, the X and Y axes of the SensorTile can be perpendicular to the gravitational field. It is critical here that the SensorTile remain absolutely level on its surface during motion.



In order to test, you will find that the SensorTile will slide smoothly on a wooden desk surface or on a large paper sheet that is also level and smooth on a surface.

Motion for this demonstration may be just 4 to 6 inches (10 to 15 centimeters).

After viewing the videos, perform these tests with your SensorTile.

[Video of SensorTile Displacement](#)

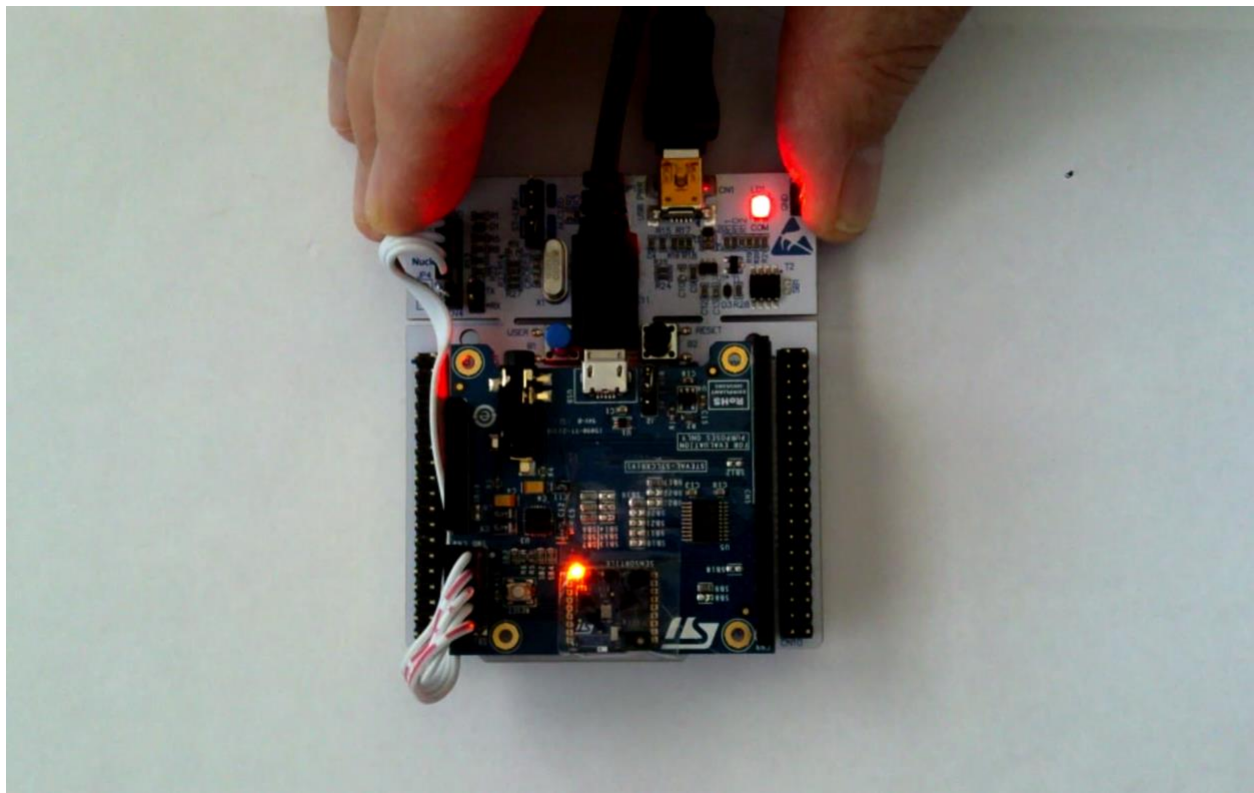


Figure 4. The image above is a hyperlink to a YouTube video. This video shows displacement of the SensorTile in first the +Y Direction (and a return to the origin) and then in the +X direction (and a return to the origin).

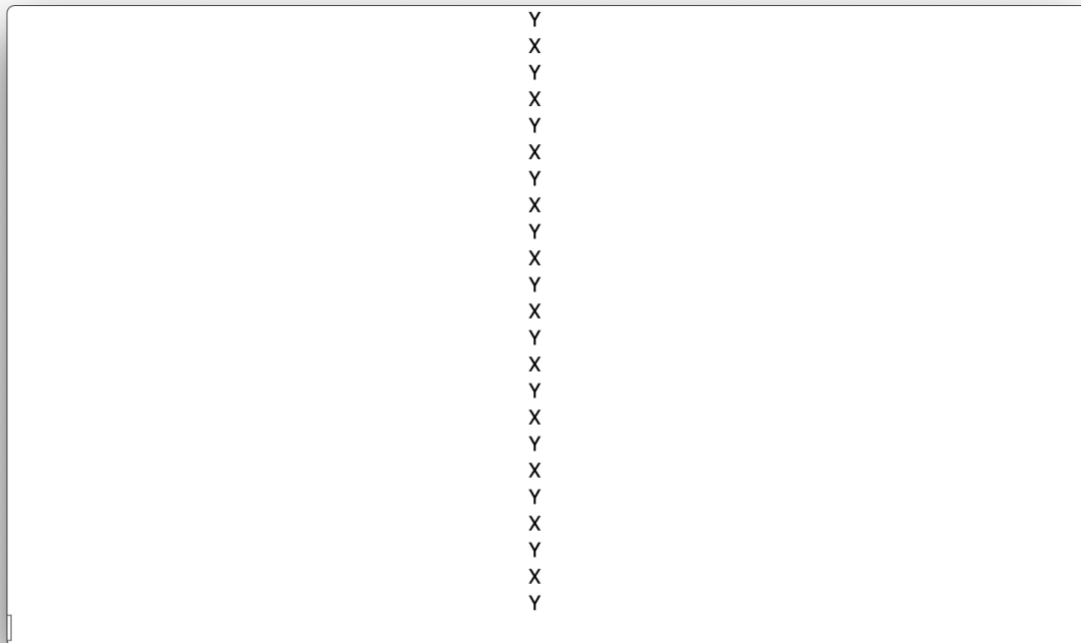


Figure 5. Real time display of computed displacement. The X symbol indicates X-Axis displacement. The Y symbol indicates Y-Axis displacement. This animation shows the results of an X-axis displacement in the negative X-Axis direction followed by an X-axis displacement in the positive X-Axis direction.



[Video of Real Time Display of -X and +X Displacement](#)

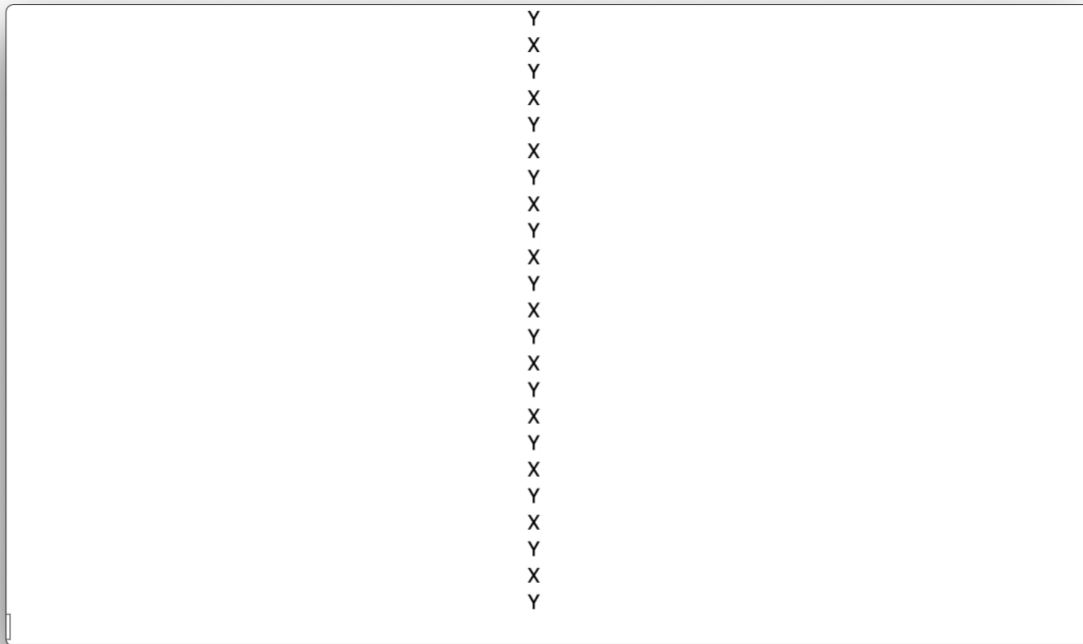


Figure 6. Real time display of computed displacement. The X symbol indicates X-Axis displacement. The Y symbol indicates Y-Axis displacement. This animation shows the results of a Y-axis displacement in the positive Y-Axis direction followed by a Y-axis displacement in the negative Y-Axis direction.

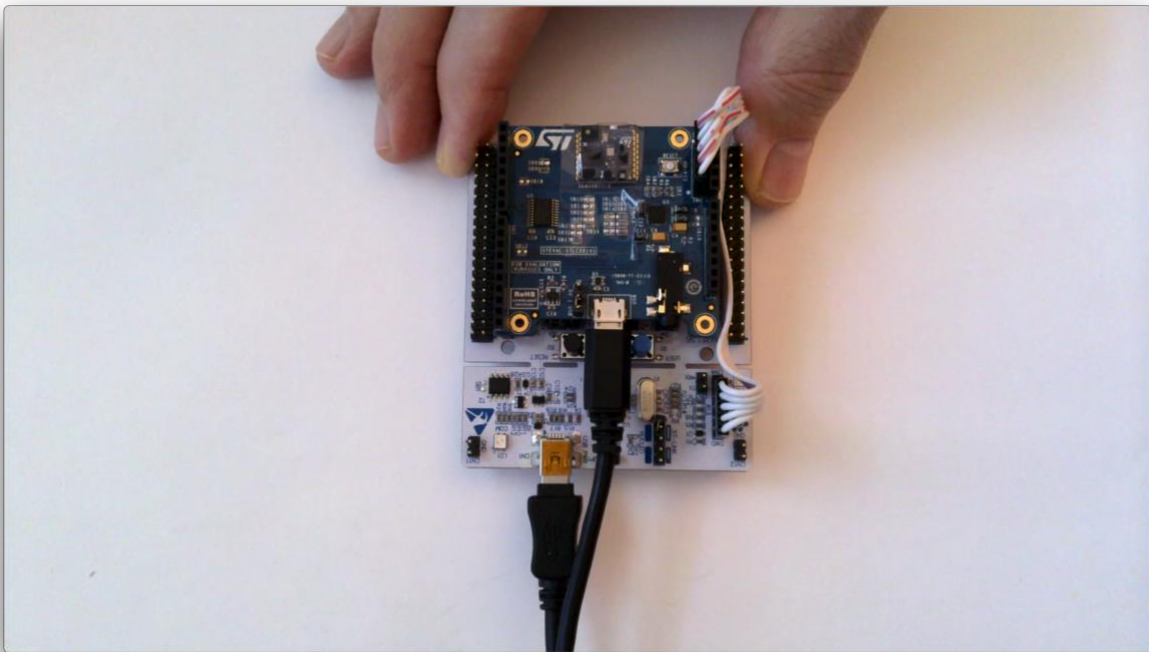


12. Gravitational Acceleration and Displacement Measurement

In this next test, the SensorTile will remain motionless. However, its angle relative to level will be changed to evaluate the effect of the gravitational acceleration vector projected on X and Y axes.

After viewing the videos, perform these tests with your SensorTile.

[Video of Tilt Motion of SensorTile from Level to +X Axis inclined upward then from Level to +Y Axis inclined upward](#)





[Video of Screen Display of Tilt of SensorTile from Level to +X Axis inclined upward then from Level to +Y Axis inclined upward](#)

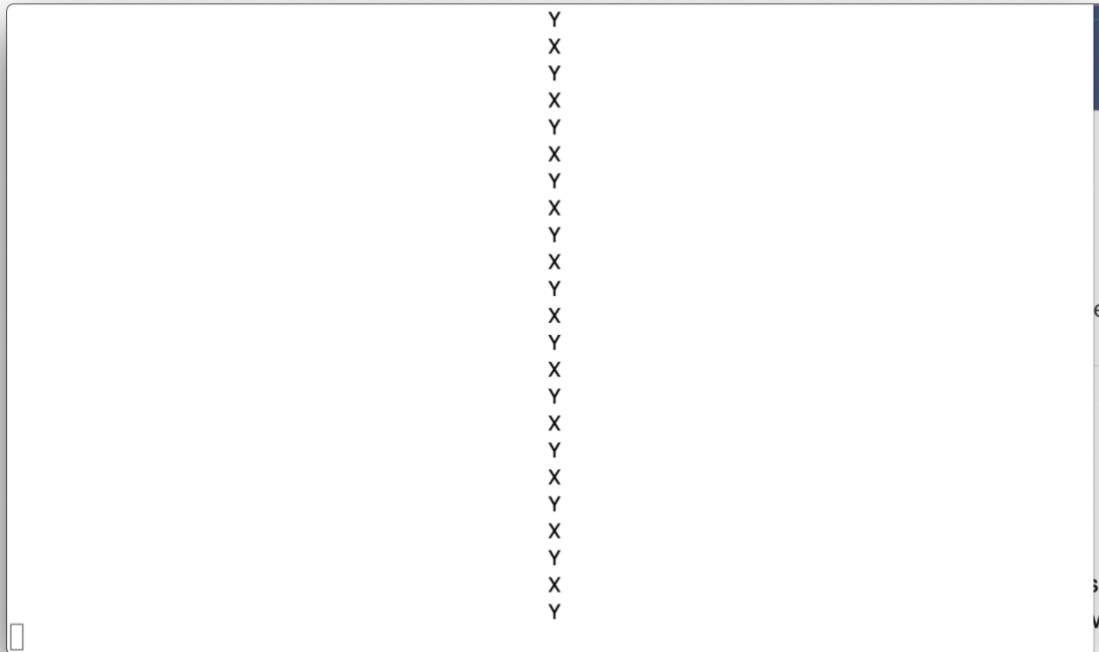


Figure 7. Real time display of computed displacement. The X symbol indicates X-Axis displacement. The Y symbol indicates Y-Axis displacement. This animation shows the results of a tilt motion from level to X-inclined upward returning to level then a tilt motion from level to Y-axis inclined upward and returning to level.



13. Assignment: Motion Sensing Filter Design

The influence of even small inclination angle values will produce an acceleration resulting in large apparent displacement. These displacements, over time, will exceed that of gesture motions.

The high pass filter, attenuates signals with frequency components above the corner frequency set by `HIGH_PASS_FILTER_FREQUENCY`. Thus, the effects of drift are reduced.

In this section, the value of this corner frequency is varied to demonstrate the effects of drift removal, and also the loss of information regarding steady state displacement.

1. Navigate to the Debug window of the STM32CubeIDE. Perform a Terminate and Remove of the task shown in the Debug window.
2. Then examine the code in `main.c` and change the `HIGH_PASS_FILTER_FREQUENCY` from 0.2 Hz to 0.02 Hz.
3. Then, with the SensorTile motionless and level, observe the X- and Y-axis acceleration signals. Note the drift in the signals.
4. Then, perform the same motions you had performed previously. Note that indications of accurate displacement are degraded.
5. Now change the `HIGH_PASS_FILTER_FREQUENCY` from 0.02 Hz to 0.002 Hz and evaluate again.

It is important to learn from this in preparation for design of systems that will apply machine learning for recognition of motion patterns, as will occur in next Tutorials.