FTF FREESCALE TECHNOLOGY FORUM 2015

Hands-On Workshop: Sensor Data Collection and Mining: **Mining Individual Sensor** Data in the Frequency-Domain + Mining Data Using Sensor Fusion, Part 3 of 3

FTF-INS-F1222

Rod Borras | Freescale AMR FAE
Michael Steffen | Freescale AMR FAE

J U N E . 2 0 1 5

freescale™

External Use

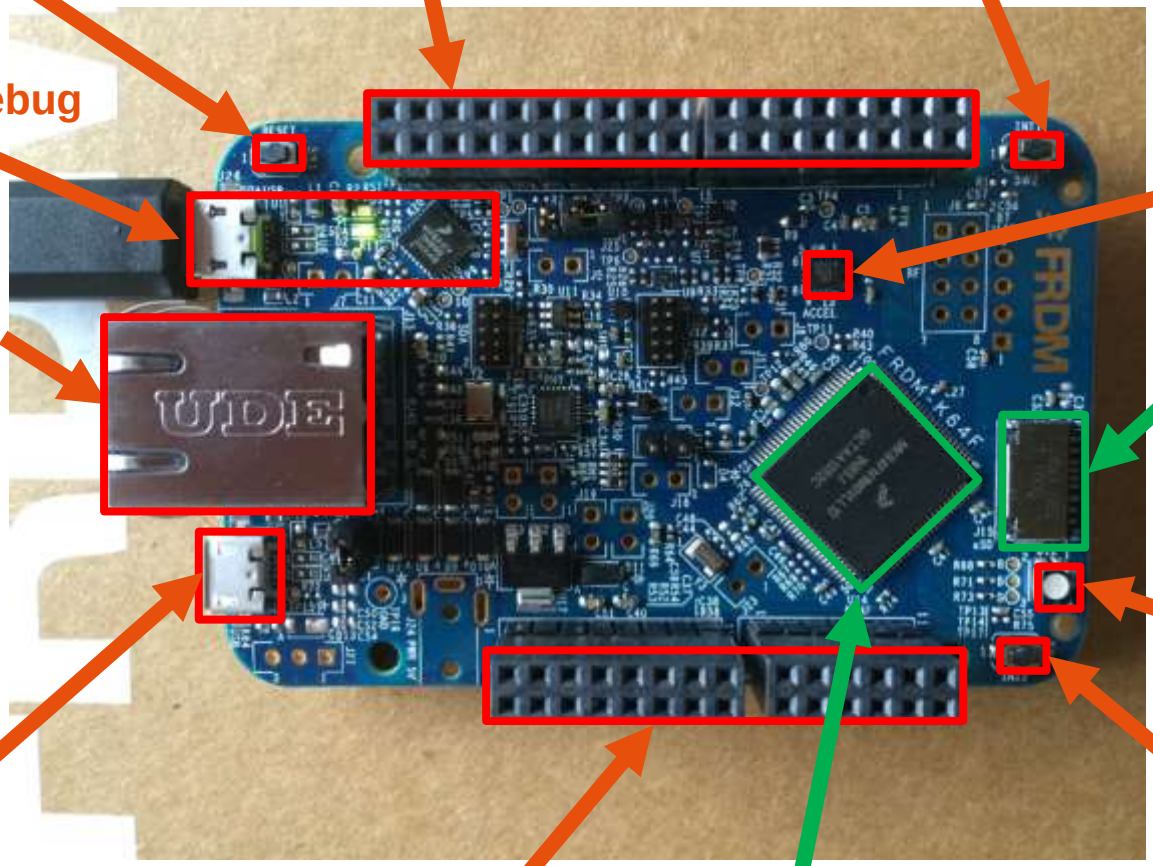
Agenda

- Attendees will learn how to capture data in the frequency domain, using real world stations in class
- How is the Frequency Domain datalogger used?
- How is the Sensor Domain datalogger used?
- Hands-on: Collect data at the different events at each station
- Hands-on: How to use the Java custom GUI to create, test, and validate your algorithms
- Hands-on (optional): How to download and test your algorithms on the real hardware
- Discussion of the top ten do’s and don'ts
- Sensor strengths and weaknesses
- Q&A

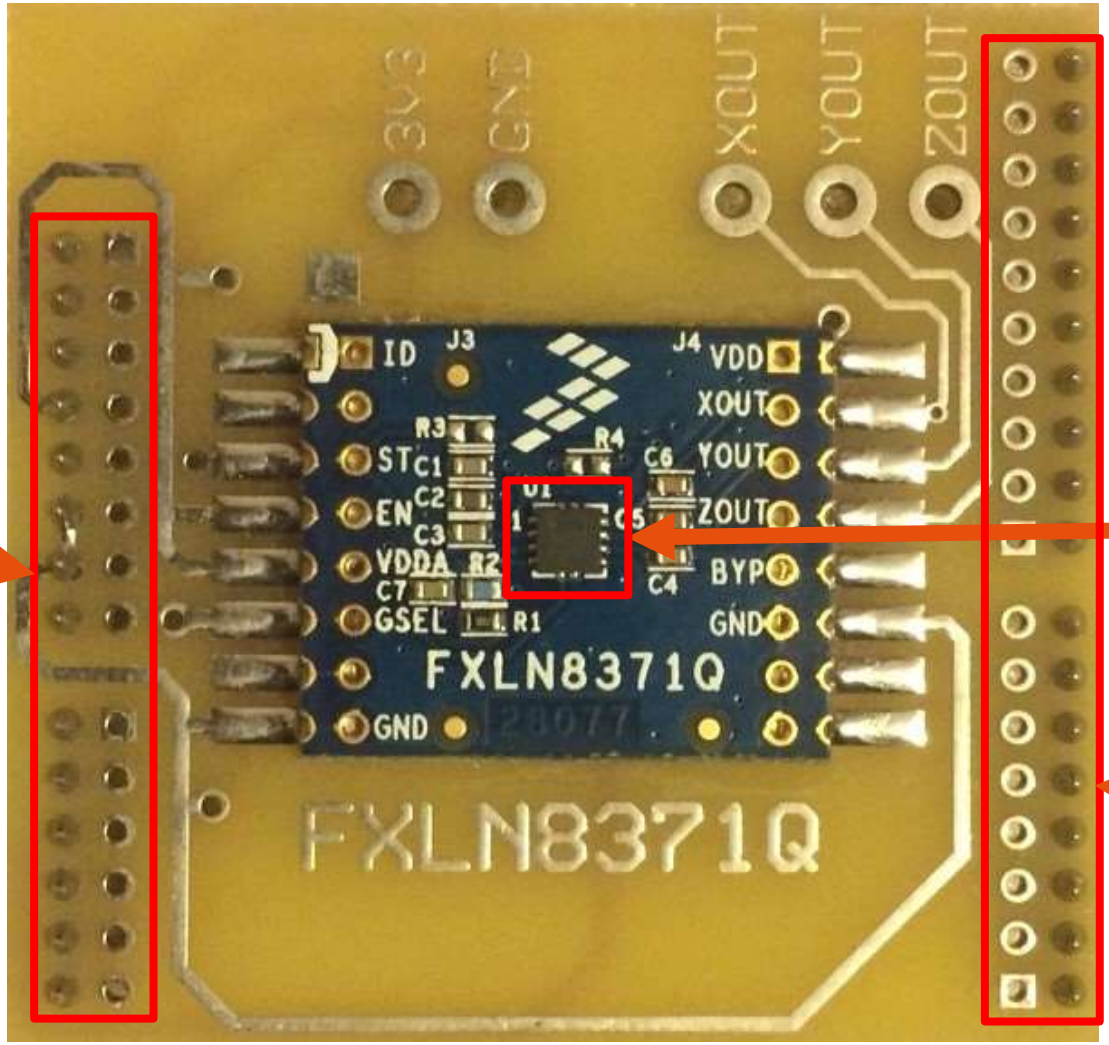
Hands-on: Collect data in the frequency domain, and with Sensor Fusion, then test and validate algorithms with custom Java GUI.

- **Lab objective 1 – Review steps involved to collect data at different stations pertaining to the frequency domain, and Sensor Fusion**
- **Lab objective 2 – Collect data at the stations and move to PC**
- **Lab objective 3 – Test your algorithms using the custom Java GUI!**

FRDM-K64F Hardware Overview



BRKOUT-FXLN8371Q Hardware Overview used in the Frequency Domain Labs

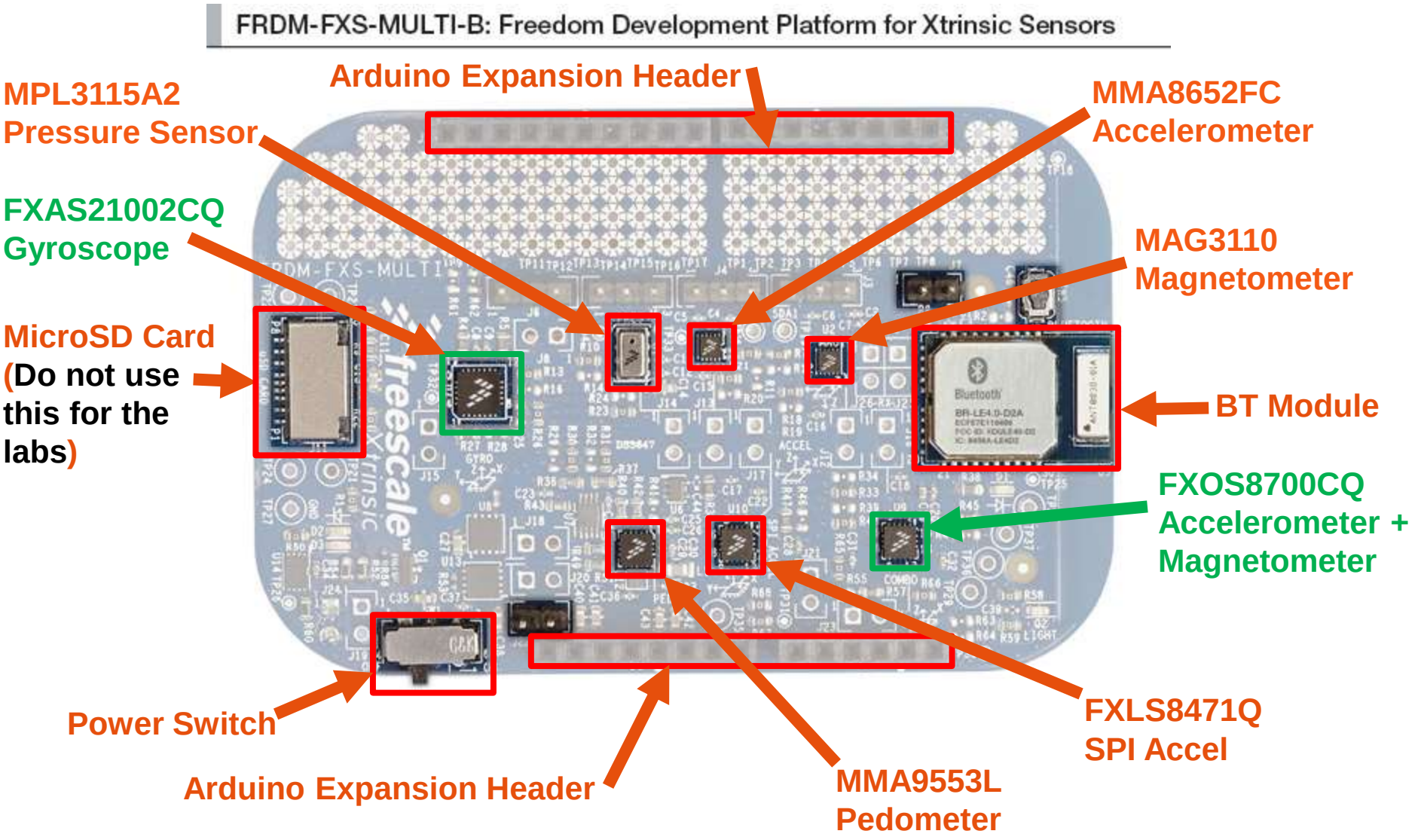


Arduino Expansion Headers

FXLN8371Q Analog Accelerometer

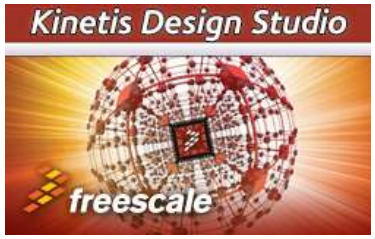
Arduino Expansion Headers

FRDM-FXS-MULTI-B Hardware Overview

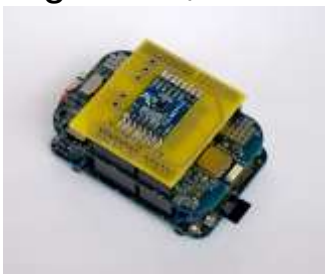


Why are we using a Java GUI approach????

- Traditional approach for algorithm development
  - Write code in Microcontroller IDE environment
  - Take time to write code, compile code, and download into a hardware development board
  - Hard to repeat exact same steps for trial and error
  - Can't re-use data files



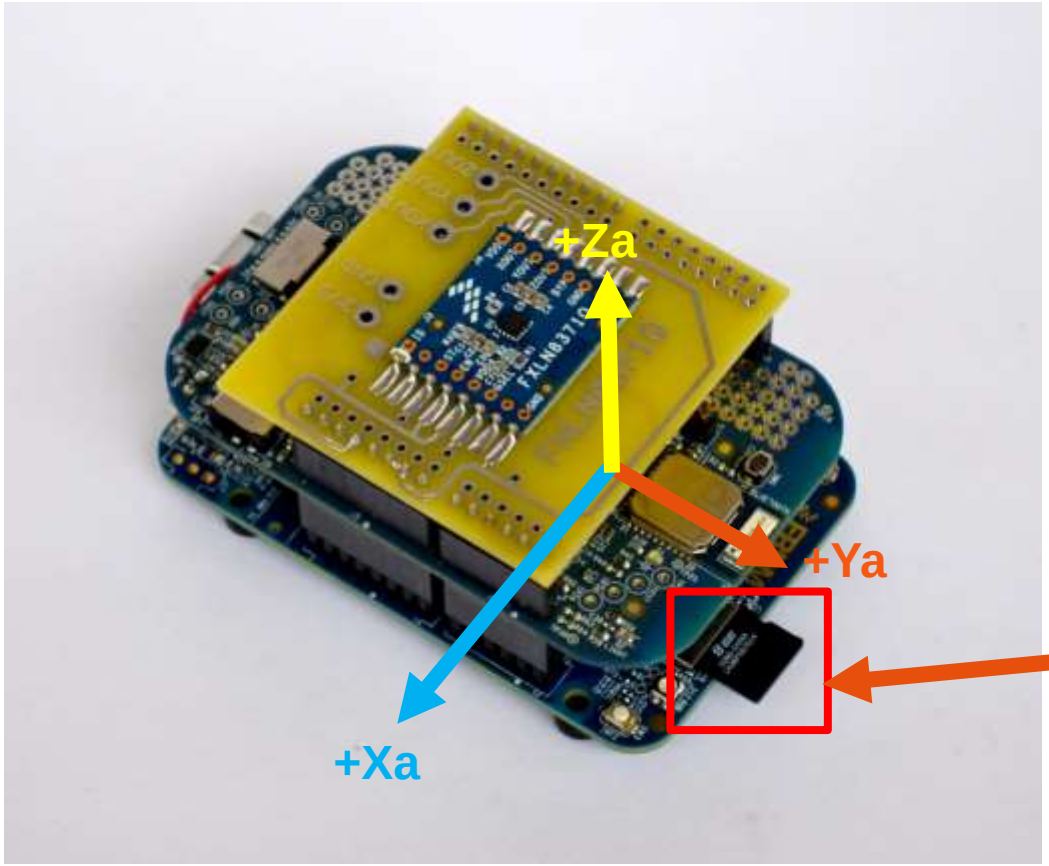
- Our approach to algorithm development
  - Program board as datalogger and then re-use same exact board to try out code
  - Ability to use the exact same datalog file again and again, test repeatability
  - Use Java GUI to develop algorithms and try out multiple algorithms
  - Easy to compare algorithms
  - When satisfied with algorithm, download back into board



HARDWARE Board Description

Hardware Datalogger Board for the **FREQUENCY DOMAIN LABS**:

- FRDM-K64F board + FRDM-FXS-MULTI-B + BRKOUT-FXLN8371Q



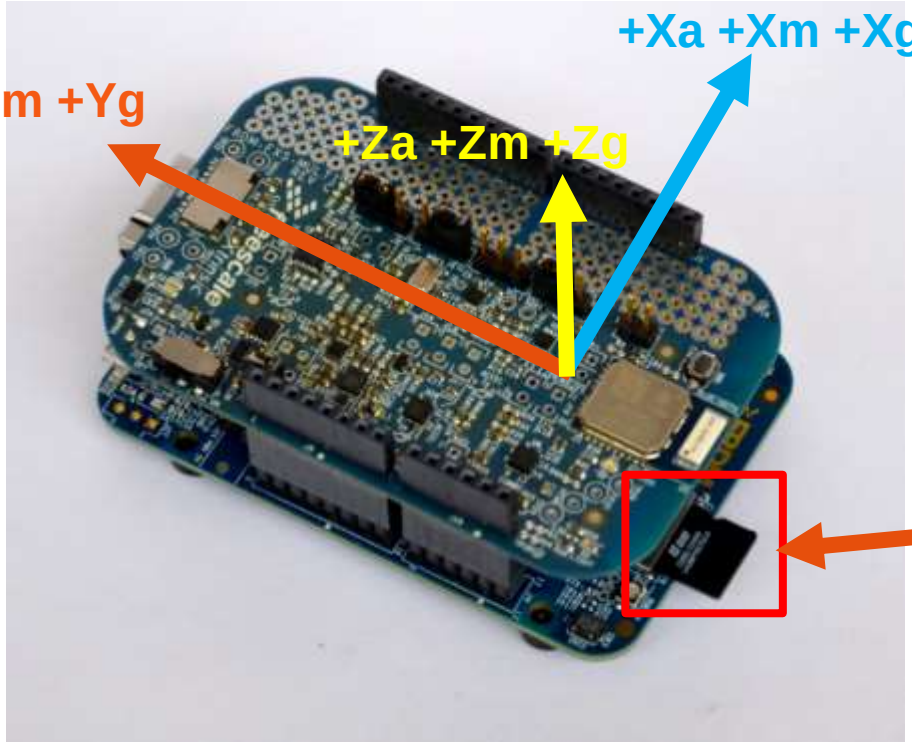
Sensor AXIS orientation
a = Accelerometer

SD Card Slot used for datalogging

HARDWARE Board Description

Hardware Datalogger Board for the **SENSOR FUSION LABS:**

- FRDM-K64F board + FRDM-FXS-MULTI-B



All Sensors AXIS orientation
a = Accelerometer
m = Magnetometer
g = Gyroscope

SD Card Slot used for datalogging

Labs - General Flow of the FREQ AND SFUS Domains Labs

- Here is the general flow of the labs:
  1. Take board to datalogging station and datalog that event

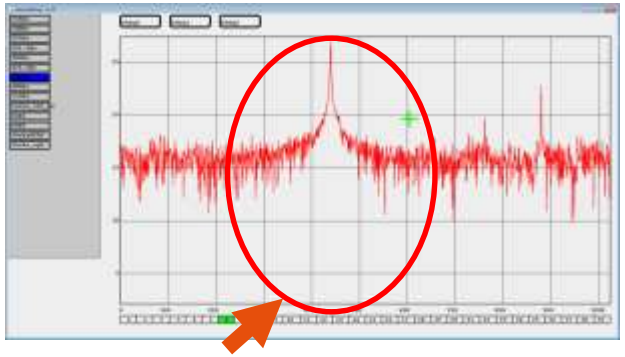
Tuning Fork Station

Motor Imbalance Station

Compass Station

Static / Linear Acceleration Station

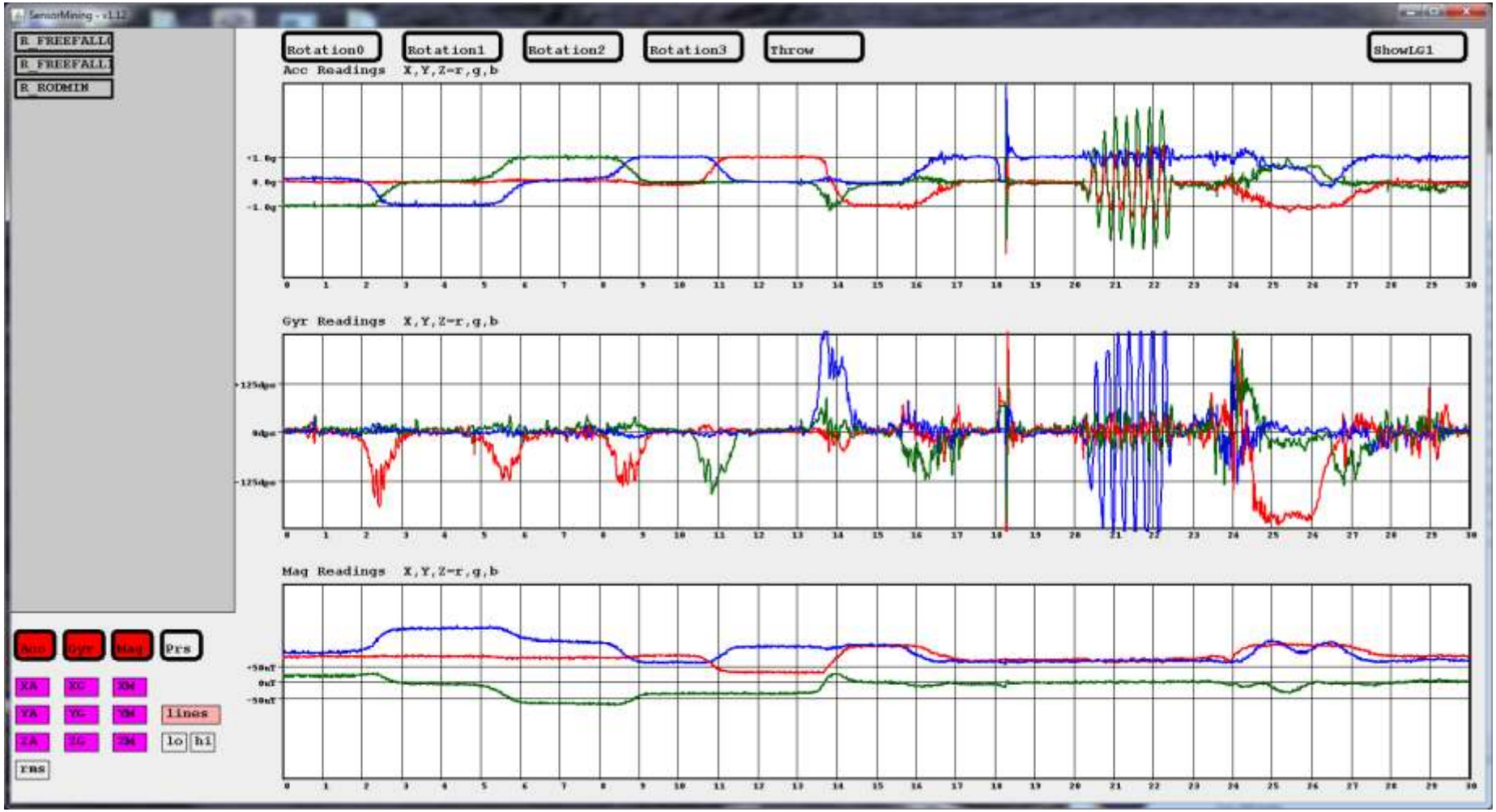
  2. Turn on datalogger and log the specific station for 30 seconds (must let datalogger time out)
  3. Take datalogger board back to desk and use the “Sensor Mining” Java GUI to analyze data:



Example FFT from 440Hz Tuning Fork Event from Station Displayed Graphically

Analyze the pre-written tuning fork algorithms or try on of your own!

Sensors 101 – Explanation of Sensors used in class



Sensor Mining Data in the Frequency Domain

SOFTWARE Description for the Frequency Domain Labs

In all the labs we will use a custom Java based GUI tool that will be referred to as “Sensor Mining Java GUI”.

A description of the opening screen:

Long Term Datalogger files Raw Values Used in Class 1 of 3

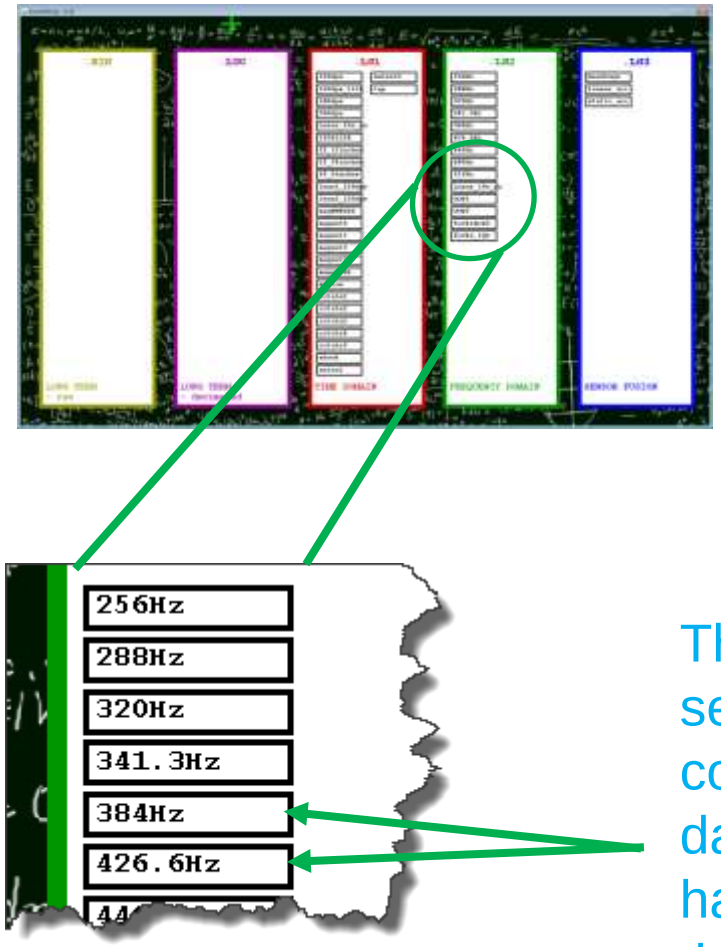
Long Term Datalogger files Pre-filtered values Used in Class 1 of 3

Time Domain Datalogger files Used in Class 2 of 3

Frequency Domain Datalogger files Used in Class 3 of 3

Sensor Fusion Datalogger files Used in Class 1 of 3

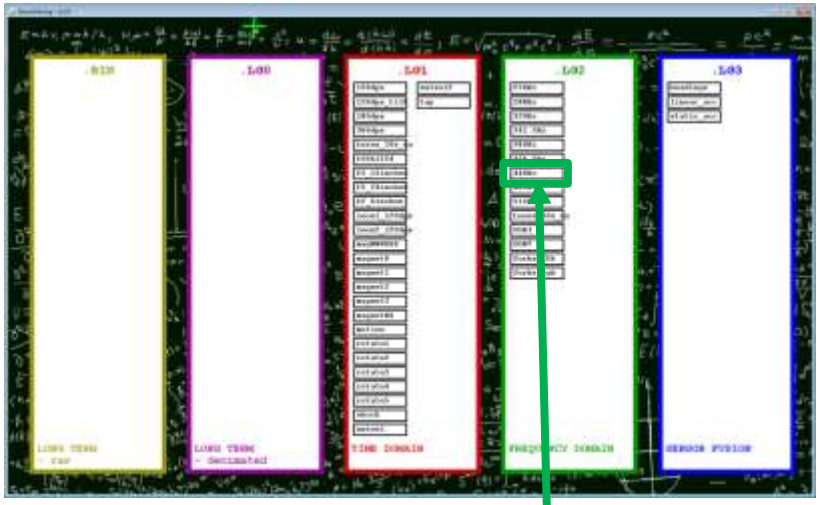
SOFTWARE Description for the Frequency Domain Labs

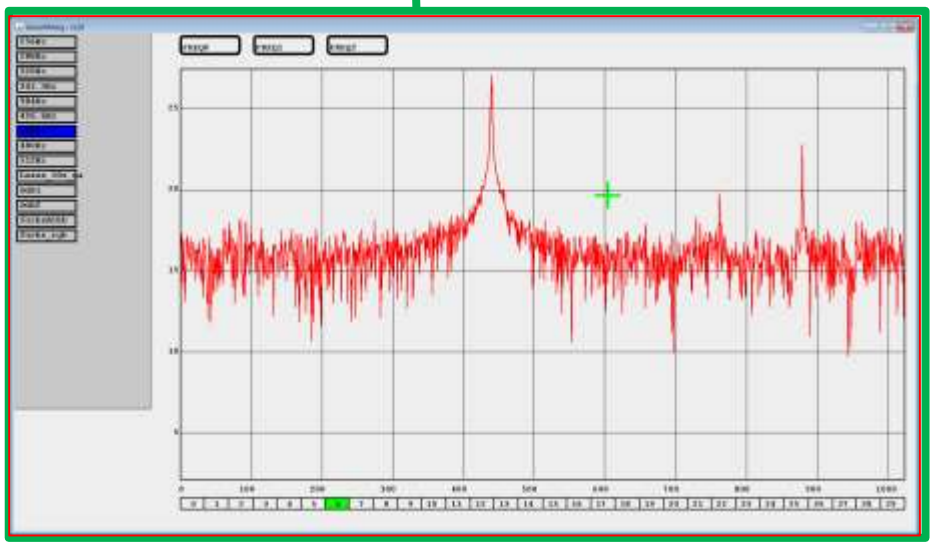


Sensor Mining Java GUI
"Frequency Domains"

These are the 30 second files collected on the datalogger hardware at each of the stations

SOFTWARE Description for the Frequency Domain Labs



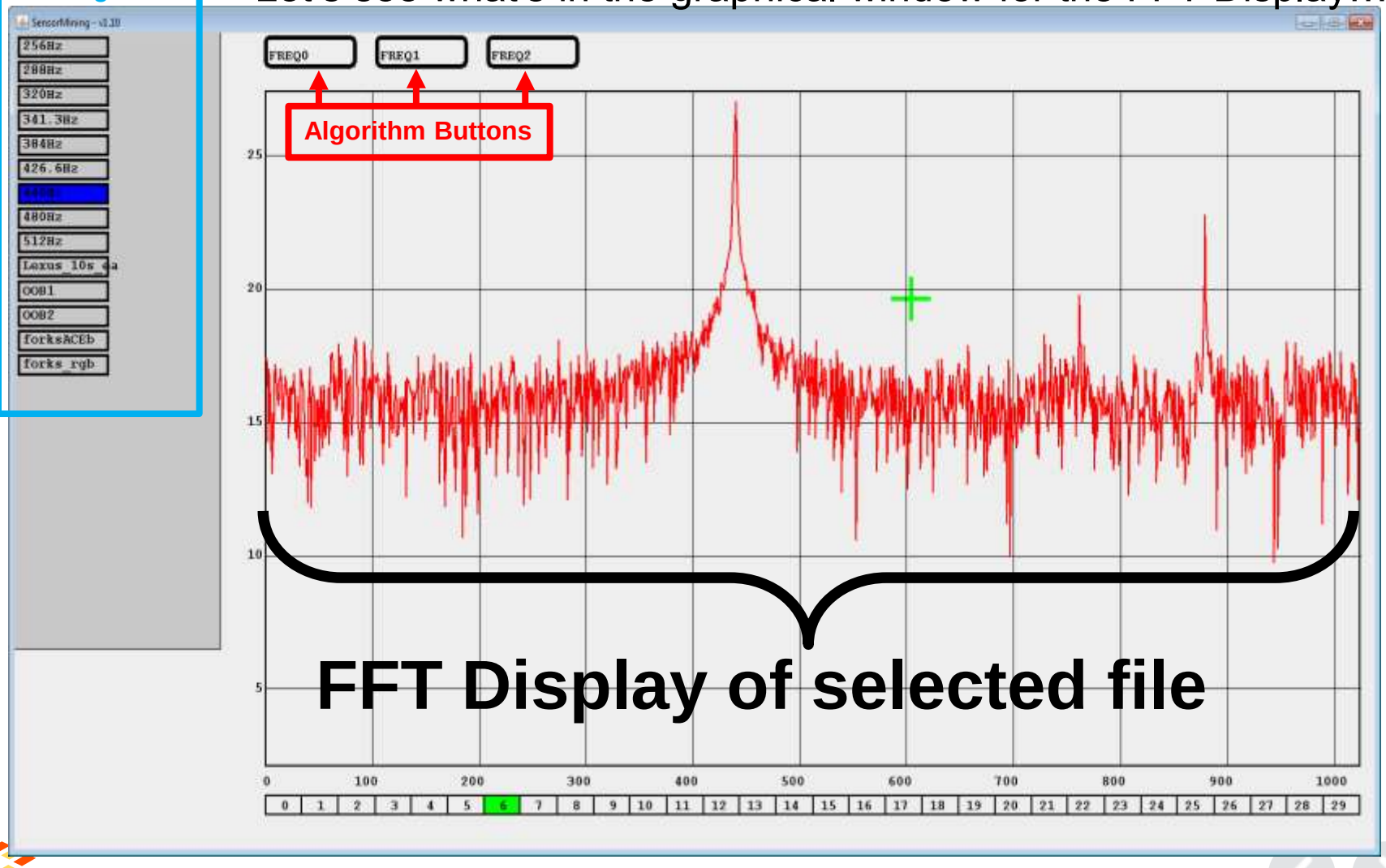


When you click on the file, you open the graphical representation window…

SOFTWARE Description for the Frequency Domain Labs

Datalog Files

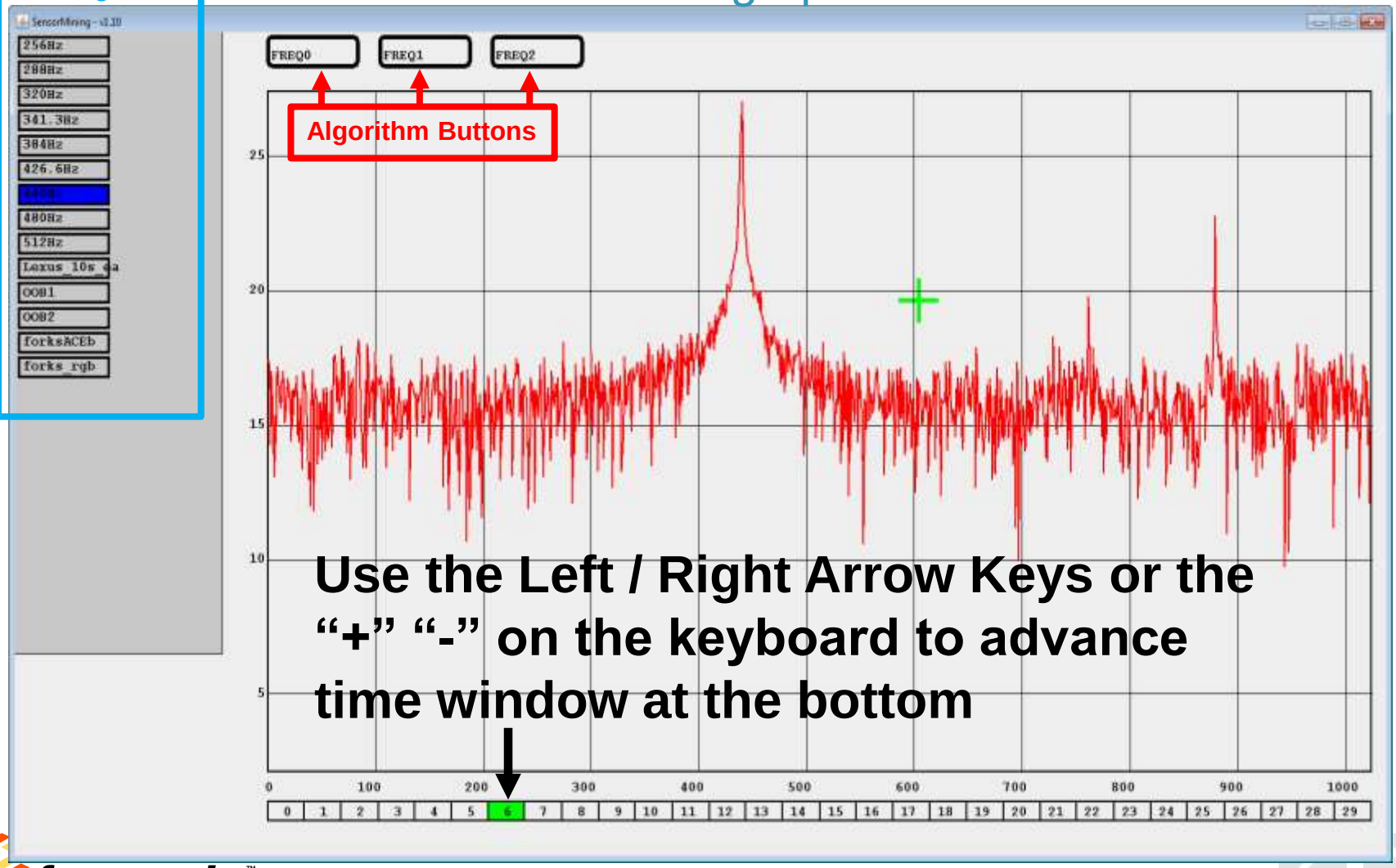
- Let's see what's in the graphical window for the FFT Display…



SOFTWARE Description for the Frequency Domain Labs

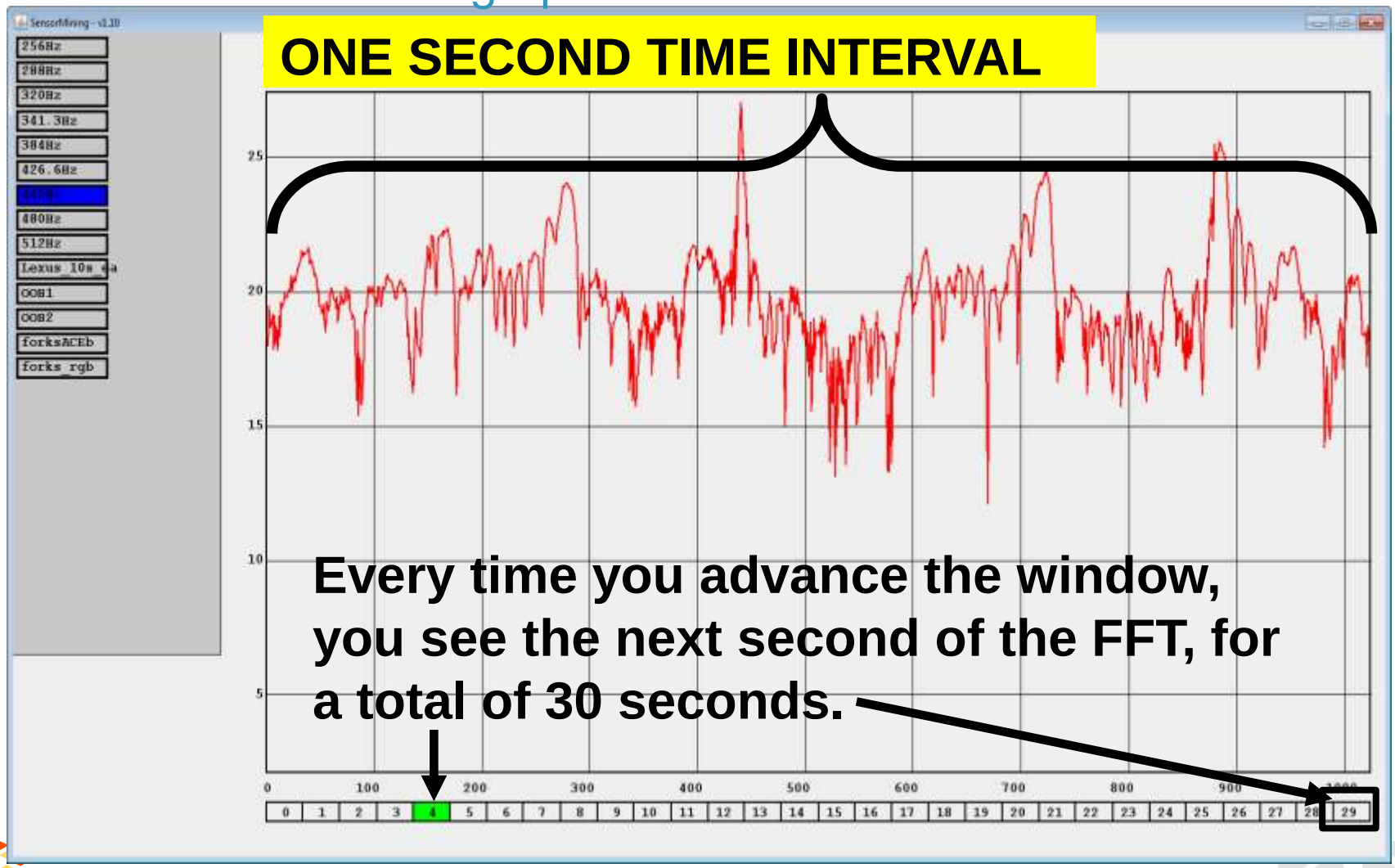
Datalog Files

- Let's see what's in the graphical window…



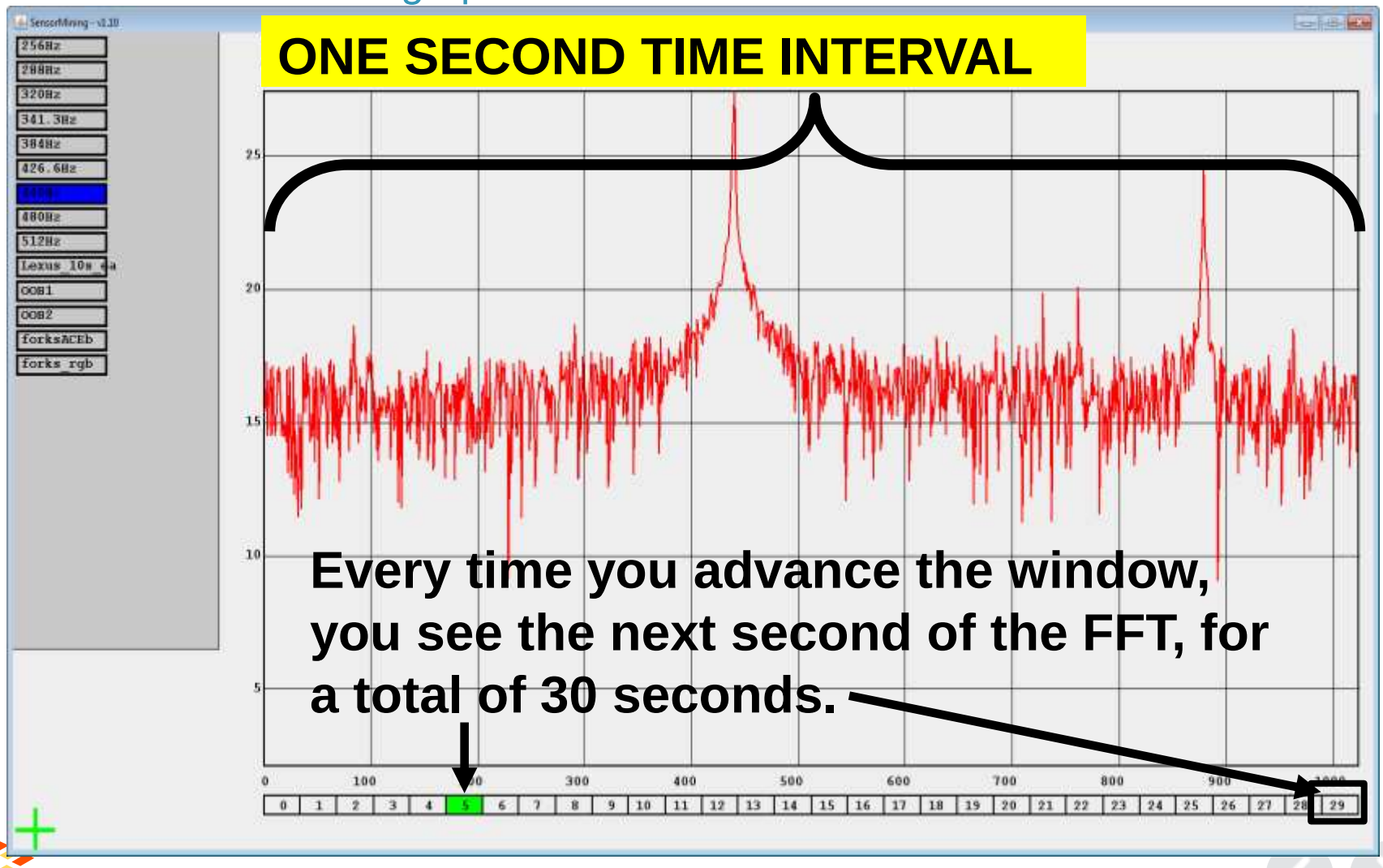
SOFTWARE Description for the Frequency Domain Labs

- Let's see what's in the graphical window…



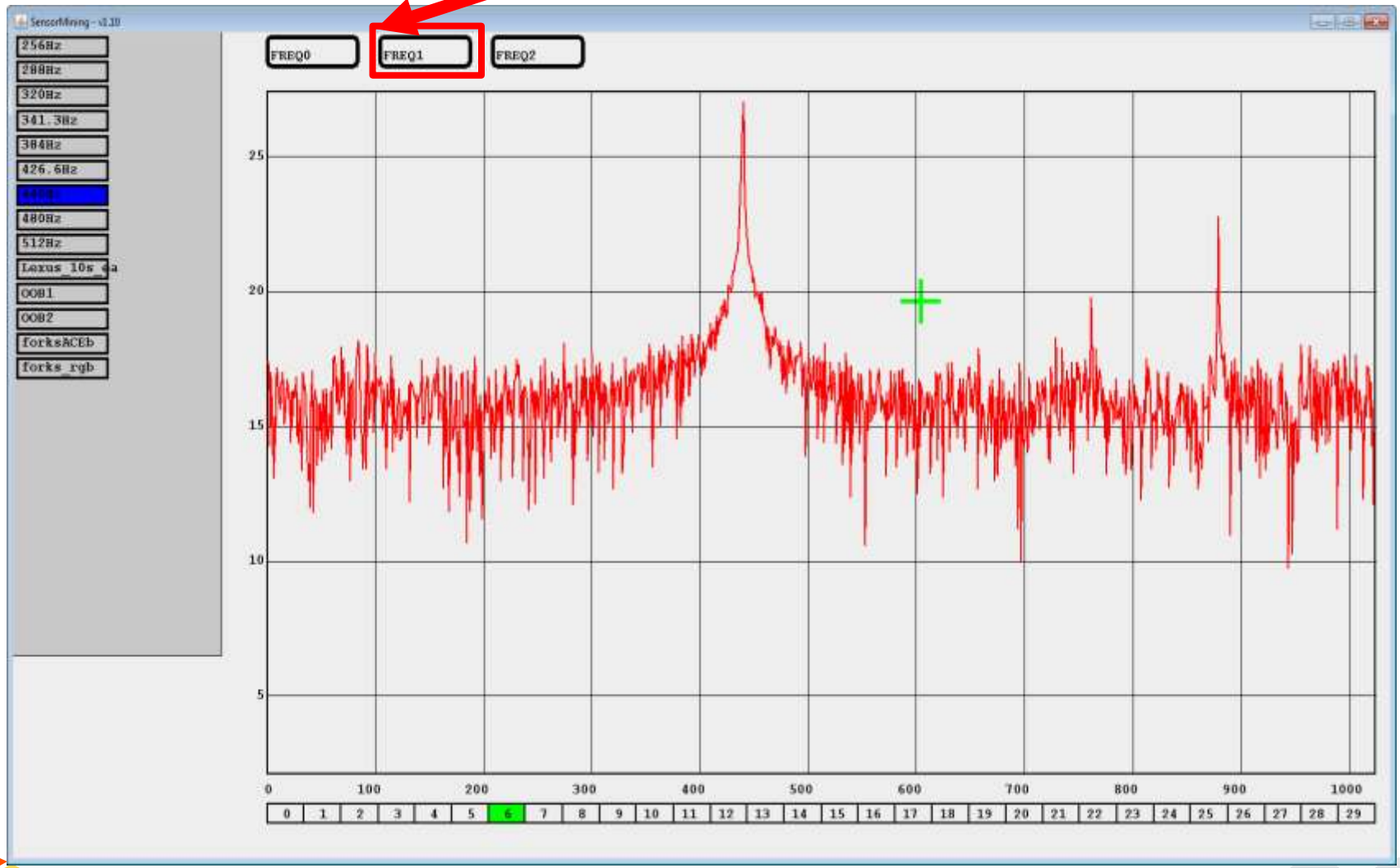
SOFTWARE Description for the Frequency Domain Labs

- Let's see what's in the graphical window…



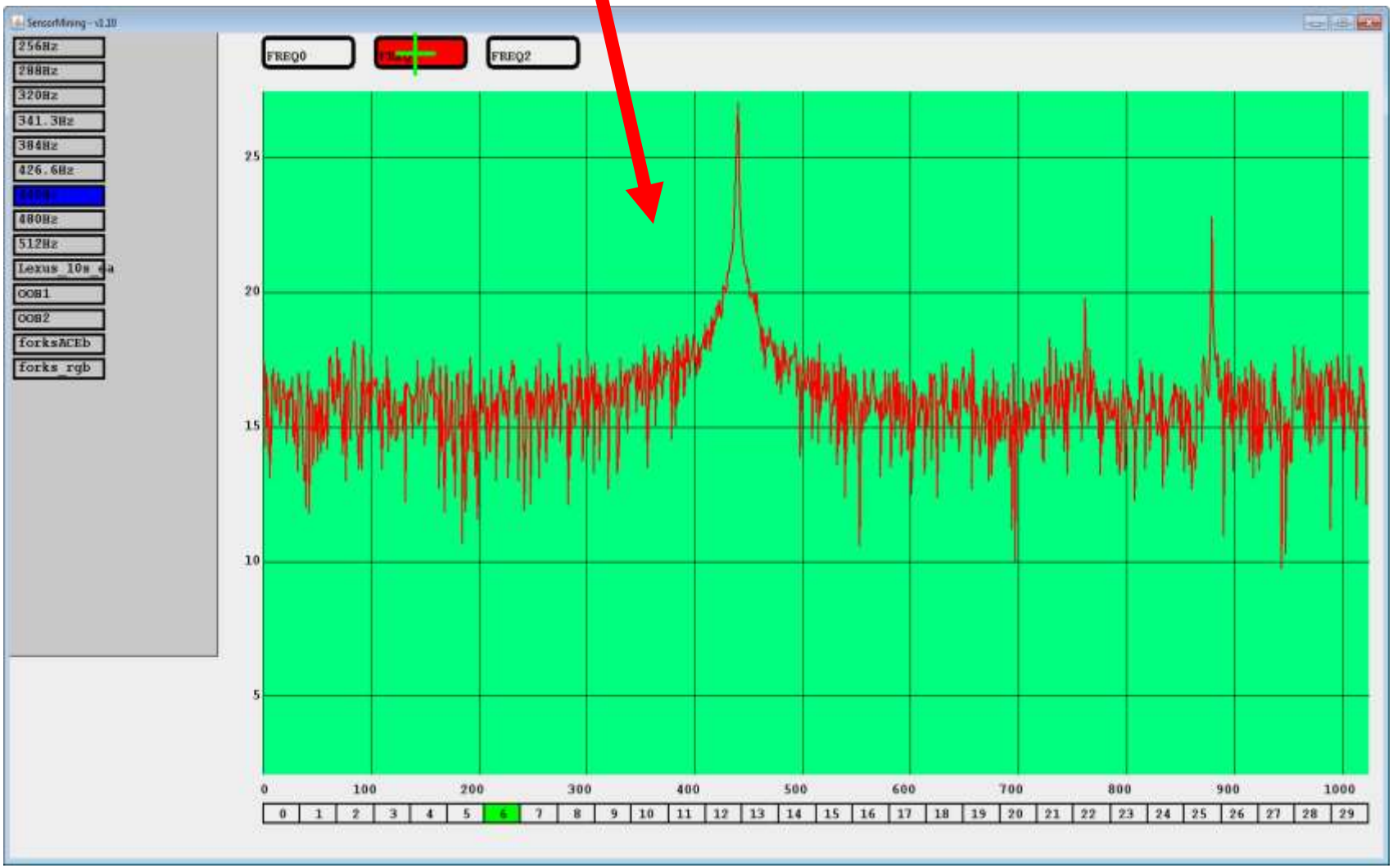
How do I apply an algorithm to my data in the Frequency Domain?

Toggle the button to apply / remove selected algorithm



The FREQ1 algorithm is applied to my data in the Frequency Domain

GREEN LED Simulation when condition is met in algorithm



Let's take a step back into the Microcontroller Embedded Code for the Frequency Domain, this is a program to perform an FFT on 30 seconds of analog sensor data.

Let explore it in detail…

```
// GLOBAL VARIABLES ////////////////////////////////////////////////
//-----------------------------------------------------------------
int fftsec;                    // FFT time window in seconds
float fft[1024];               // FFT array for each time window (0,1,2,3...29)
//-----------------------------------------------------------------

// FUNCTION HEADERS ////////////////////////////////////////////////
//-----------------------------------------------------------------
void Algorithm_init();
byte Algorithm_run();
//-----------------------------------------------------------------

//**** MAIN ************************************************************
//=================================================================
int main(void)                 // Main Loop
{
Initialize_MCU_and_Sensors();  // Initialize MCU and Sensors
Algorithm_init();              // Initialize Algorithm
for (;;)                       // Loop forever
    {
    Read_Sensors();            // Reads the all the sensors on the hardware board
    LEDcolor=Algorithm_run();  // Algorithm function run and returns LED color
  switch (LEDcolor)
    {
    case 0: RGB(0,0,0); break; // LED off
    case 1: RGB(1,0,0); break; // LED=red
    case 2: RGB(0,1,0); break; // LED=green
    case 3: RGB(0,0,1); break; // LED=blue
    }                          // for loop end
  }                            // main loop end
return 0;
}
//=================================================================
//-----------------------------------------------------------------
void Algorithm_init()
{}
//-----------------------------------------------------------------
byte Algorithm_run()
{
int res=0;
if (fft[440]>20) res=1;
return res;
//-----------------------------------------------------------------
```

Variables used for FFT function

Algorithm initialization and run function prototypes

main.c loop

Let's look at the LEDcolor switch statement:

If the function returns a number from 0-3, then a specific color turns on every time the for loop executes

Algorithm initialization (this is where you put any variables used that are not supplied above).

Algorithm function: this example states that if the frequency value at 440Hz is greater than log 20, then turn on the RED LED in the (0-29) time second windows. *So, this algorithm is detecting a turning fork at 440Hz, and turns on the RED LED.*

Let's take a look at the Microcontroller Embedded Code in the Frequency Domain

These are the variables used to create the datalog files in the Embedded Microcontroller K64F MCU, in Kinetis Design Studio…

Frequency Domain

```
// GLOBAL VARIABLES ////////////////////////////////////////////////
//-----------------------------------------------------------------
int fftsec;                 // FFT time window in seconds
float fft[1024];            // FFT array for each time window (0,1,2,3...29)
//-----------------------------------------------------------------
```

…and they happen to be the **EXACT SAME** variables used in the JAVA GUI on the hands-on labs.

Variables available to you when you write your Algorithms in the Java GUI…

Algorithms.java

```
11
12  private final int fftn=1024;
13  private int   fftsec;
14  private float fft[];
15
16  private float pit,rol,yaw;
17  private float hdg;
18  private float lax,lay,laz;
19  private float icx,icy,icz;
```

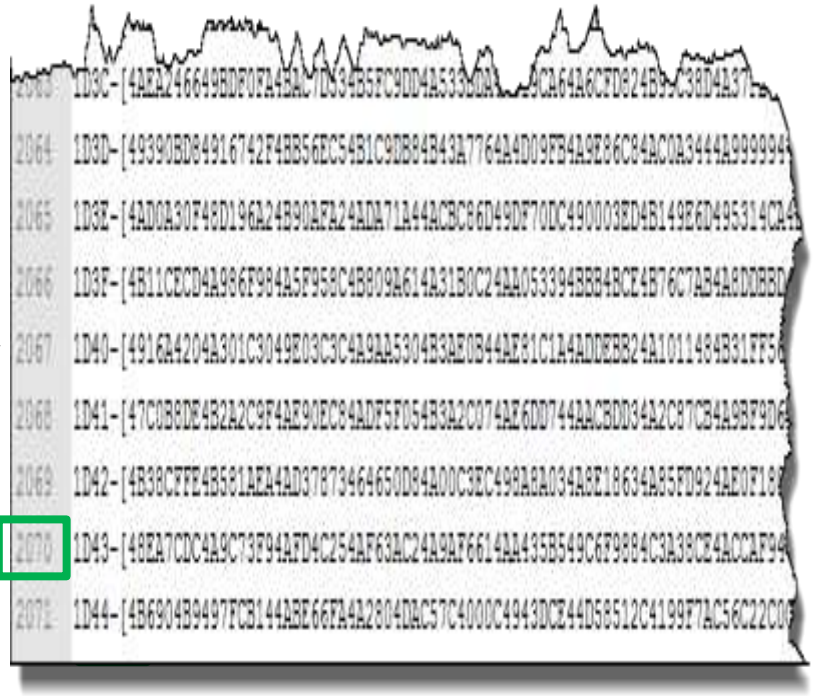
Let's compare the Java GUI to Microcontroller Embedded Code for the Frequency Domain

Embedded Microcontroller Code from the MK64F MCU. This is the code used to try your algorithm in the Java GUI on **REAL** hardware.

```
//------------------------------------------------------------------
void FourierTransform(float AngleNumerator, float *RealIn, float *ImagIn, float *RealOut, float *ImagOut)
{
int i,j,k,n,BlockSize,BlockEnd;
float delta_angle,delta_ar,alpha,beta,tr,ti,ar,ai;
for (i=0;i<fft_NS;i++) {j=ReverseBits(i); RealOut[j]=RealIn[i]; ImagOut[j]=ImagIn[i];}
BlockEnd=1; BlockSize=2;
while (BlockSize<=fft_NS)
    {
  delta_angle=AngleNumerator/BlockSize;
    alpha=sin(0.5*delta_angle); alpha=2.0*alpha*alpha;
    beta=sin(delta_angle);
  i=0;
  while (i<fft_NS)
        {
        ar=1.0; //cos(0)
        ai=0.0; //sin(0)
        j=i;
        for (n=0;n<BlockEnd;n++)
            {
            k=j+BlockEnd;
            tr=ar*RealOut[k]-ai*ImagOut[k];
            ti=ar*ImagOut[k]+ai*RealOut[k];
            RealOut[k]=RealOut[j]-tr;
            ImagOut[k]=ImagOut[j]-ti;
            RealOut[j]=RealOut[j]+tr;
            ImagOut[j]=ImagOut[j]+ti;
            delta_ar=alpha*ar+beta*ai; ai=ai-(alpha*ai-beta*ar); ar=ar-delta_ar;
            j++;
            }
        i=i+BlockSize;
        }
  BlockEnd=BlockSize;
  BlockSize=BlockSize<<1;
    }
}
```

main loop

Captured Datalog Raw file from Frequency Domain stations



Each second of FFT generates 1024 values. At 15 values per line, we need 69 lines per second. So 69 x 30sec = 2070 lines of data for the FFT.

Let's jump back into the Java GUI algorithms…assigning algorithms to GUI buttons names in Frequency Domain

- In the labs, when you write your Algorithms, you will get to assign the button names for up to ten buttons (ten algorithms) for each of the labs in the Frequency Domain in the examples below:

```
// LG2 Algorithms (FREQUENCY DOMAIN)
//
private int LG2algo0()
{
int res=0;
return res;
}
//
private int LG2algo1()
{
int res=0;
res=2;
return res;
}
//
private int LG2algo2()
{
int res=0;
res=3;
return res;
}
```

Frequency Domain Screen

LG2algo0() is given the button name "FREQ0"
LG2algo1() is given the button name "FREQ1"
LG2algo2() is given the button name "FREQ2"

How does the Java Algorithm run on the Sensor Mining GUI in the Frequency Domain?

- Here is the most important part of understanding the Sensor Mining GUI…

ALGORITHMS WILL EXECUTE 30 times when the button is pressed!

```
// LG2 Algorithms (FREQUENCY DOMAIN)
// ========================================
private int LG2algo0()
{
int res=0;
if (fft[440]>20) res=1;
return res;
}
//----------------------------------------
```

RED LED “ON” for the ENTIRE DURATION OF THE FFT TIME WINDOW.

Sensor Mining Sensor Fusion Data

SOFTWARE Description for the Sensor Fusion Labs

In all the labs we will use a custom Java based GUI tool that will be referred to as “Sensor Mining Java GUI”.

A description of the opening screen:

Long Term Datalogger files Raw Values Used in Class 1 of 3

Long Term Datalogger files Pre-filtered values Used in Class 1 of 3

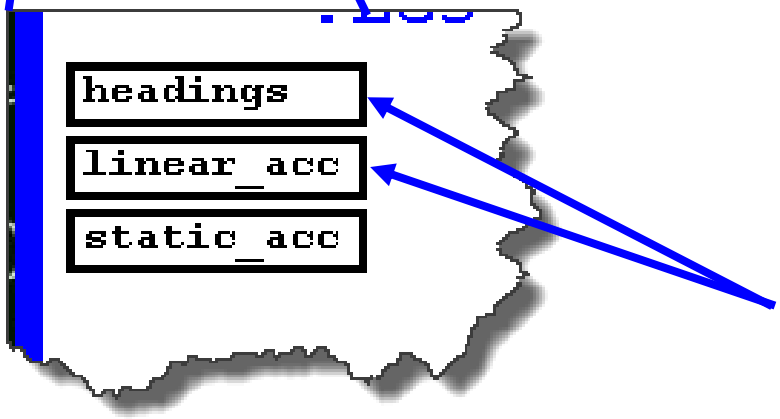
Time Domain Datalogger files Used in Class 2 of 3

Frequency Domain Datalogger files Used in Class 3 of 3

Sensor Fusion Datalogger files Used in Class 1 of 3

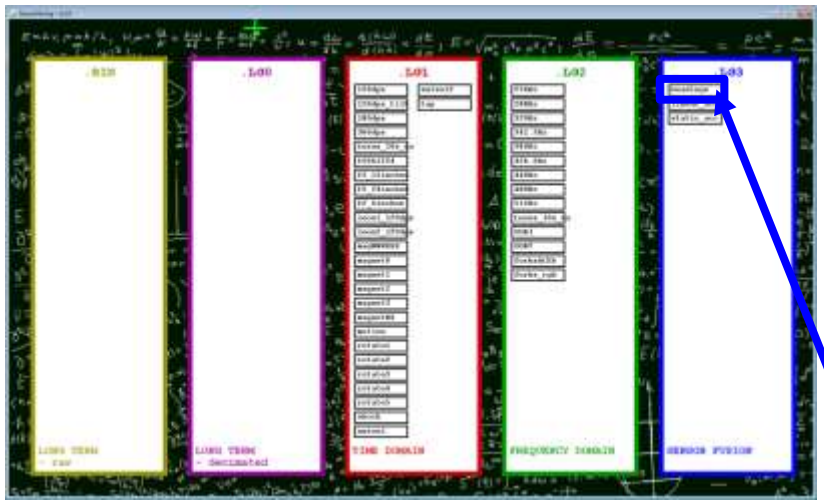
SOFTWARE Description for the Sensor Fusion Labs

Sensor Mining Java GUI "Sensor Fusion Domains"

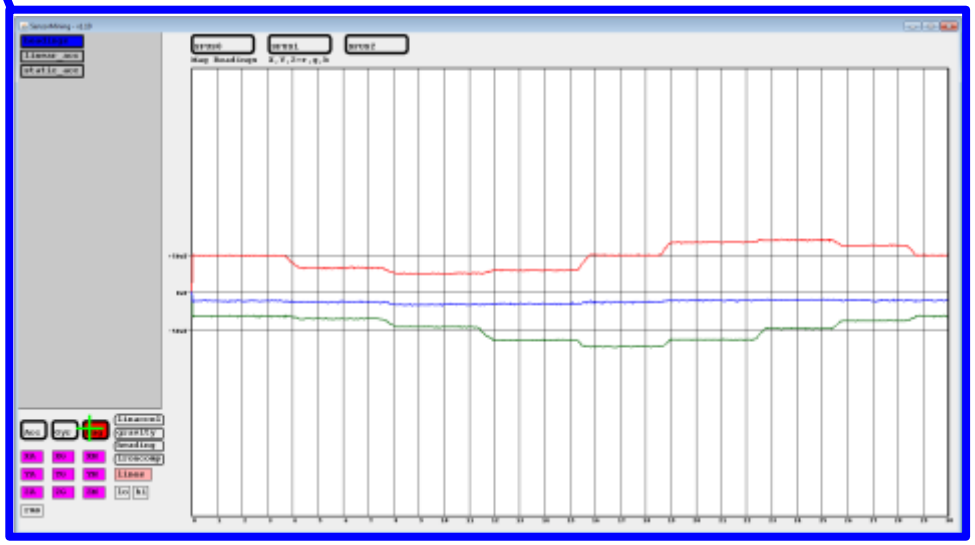


These are the 30 second files collected on the datalogger hardware at each of the stations

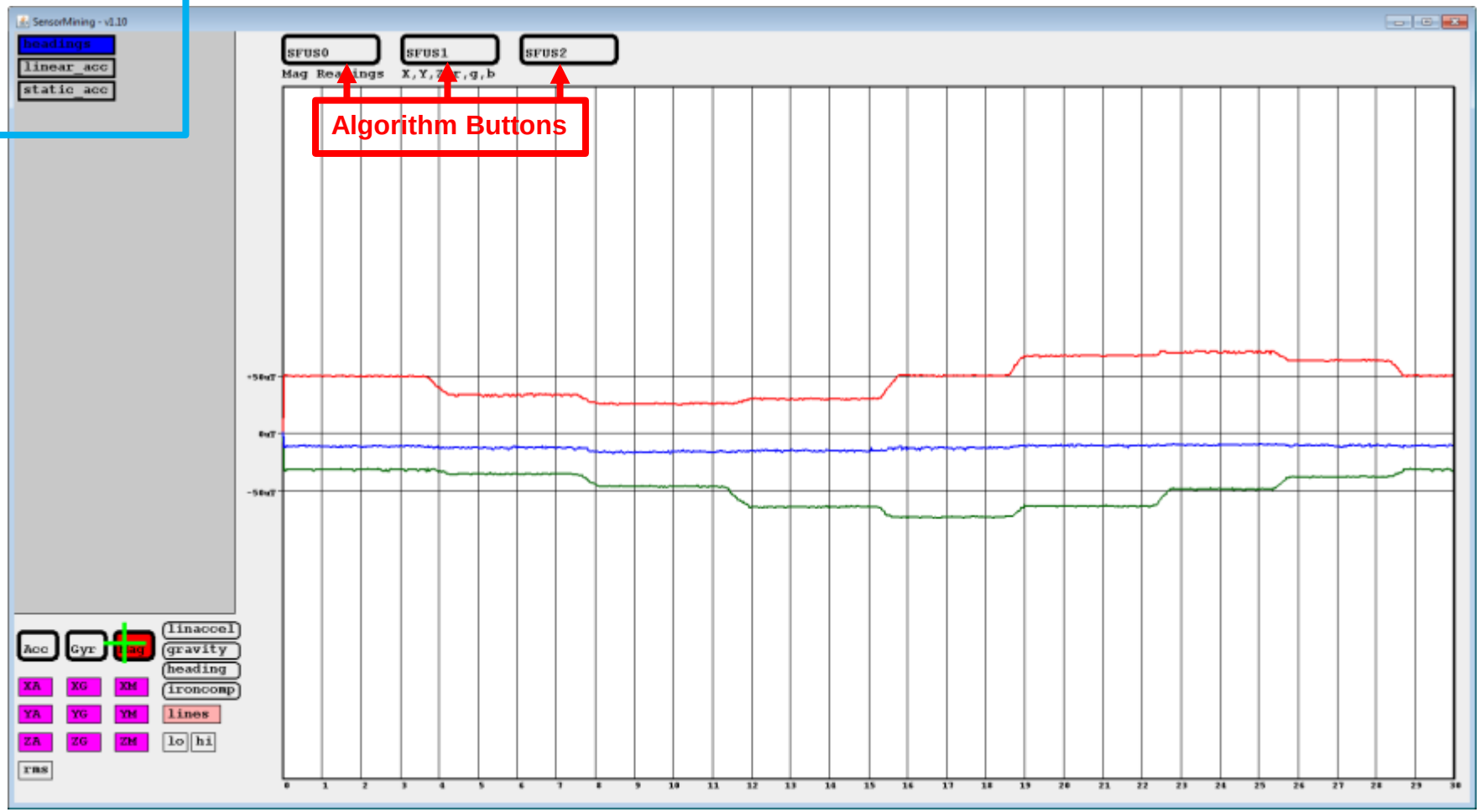
SOFTWARE Description for the Sensor Fusion Labs



When you click on the file, you open the graphical representation window…

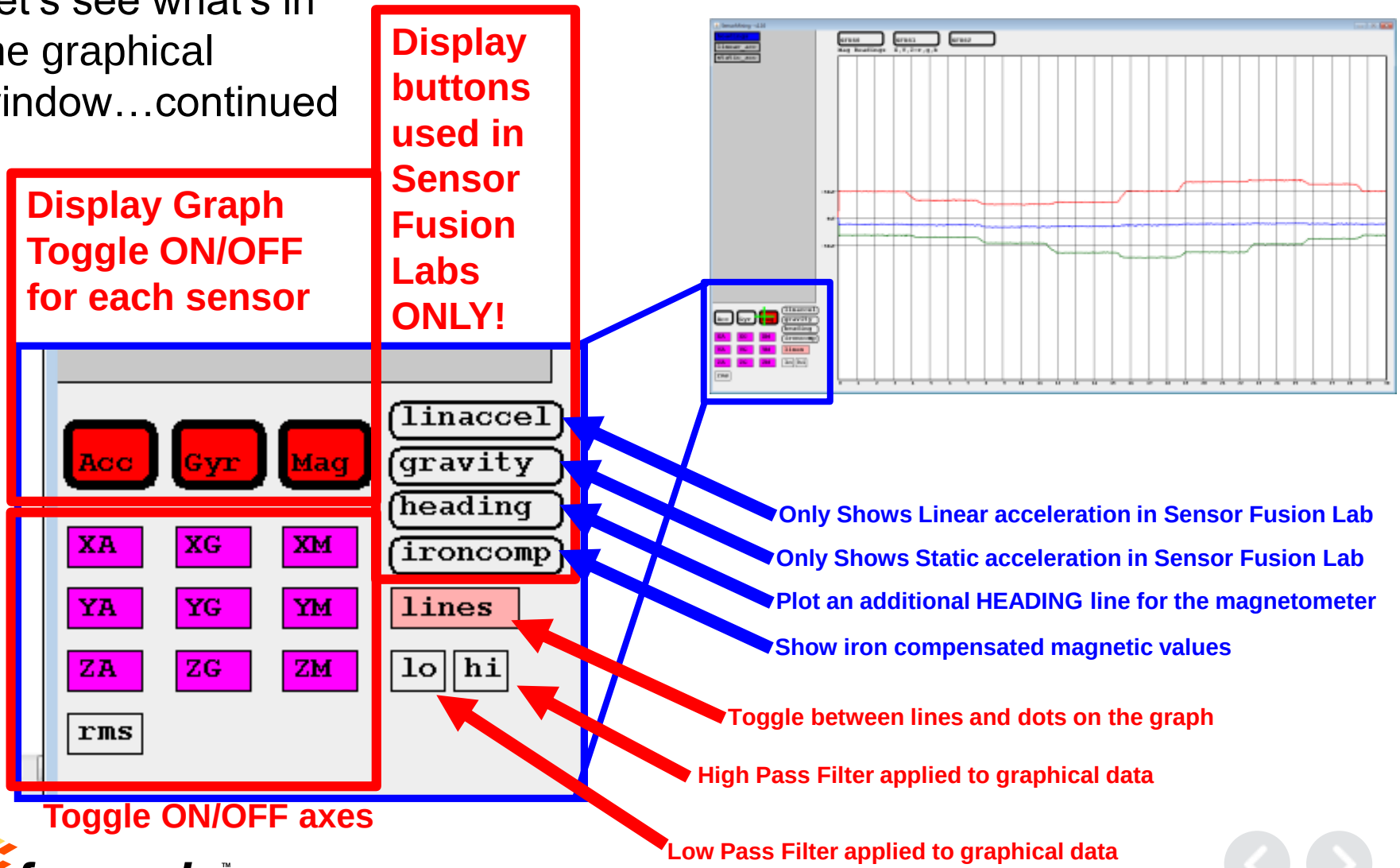


SOFTWARE Description for the Sensor Fusion Labs



SOFTWARE Description for the Sensor Fusion Labs

- Let's see what's in the graphical window…continued



The SFUS0 algorithm is applied to my data in the Sensor Fusion Domain

GREEN LED Simulation when condition is met in algorithm

Let's take a step back into the Microcontroller Embedded Code for the Sensor Fusion Domain. This program light up an LED based on compass heading. **Let's explore it in detail…**

```
void UserMediumFrequencyTaskRun(void)
{
switch (runstate)
    {
    case 0: // reset values
        seconds=0; RGB(0,0,1);
        Algorithm_init();
        runstate=1;
        break;
    case 1: // wait for calibration (give up after 20 seconds)
        if ((seconds>=20*25)||(thisMagCal.fFitErrorpc<5.0f)) {RGB(1,0,0); seconds=0; runstate=2;}
        break;
    case 2: // go Red for 5 seconds
        pit=thisSV_9DOF_GBY_KALMAN.fThePl;    // pitch
        rol=thisSV_9DOF_GBY_KALMAN.fPhiPl;    // roll
        yaw=thisSV_9DOF_GBY_KALMAN.fPsiPl;    // yaw
        hdg=thisSV_9DOF_GBY_KALMAN.fRhoPl;    // heading
        lax=thisSV_9DOF_GBY_KALMAN.faSePl[0]; // linXA
        lay=thisSV_9DOF_GBY_KALMAN.faSePl[1]; // linYA
        laz=thisSV_9DOF_GBY_KALMAN.faSePl[2]; // linZA
      icx=thisMag.fBc[0];                   // calXM
        icy=thisMag.fBc[1];                   // calYM
        icz=thisMag.fBc[2];                   // calZM
        LEDcolor=Algorithm_run();
      switch (LEDcolor)
        {
        case 0: RGB(0,0,0); break;  // LED off
        case 1: RGB(1,0,0); break;  // LED=red
        case 2: RGB(0,1,0); break;  // LED=green
        case 3: RGB(0,0,1); break;  // LED=blue
        case 4: RGB(1,1,0); break;  // LED=yellow
        case 5: RGB(1,0,1); break;  // LED=magenta
        case 6: RGB(0,1,1); break;  // LED=cyan
        case 7: RGB(1,1,1); break;  // LED=white
        case 8: RGB(0,0,0); break;  // LED=off
        }
        break;
    }
}
```

The Sensor Fusion code is a bit more complex. All the user functions are written in a routine called "*UserMediumFrequencyTaskRun()*" This task is called every 25Hz.

Let's look at the LEDcolor switch statement:

If the function returns a number from 0-8, then a specific color turns on every time the for loop executes

Let's take a look at the Microcontroller Embedded Code in the Sensor Fusion Domain

These are the variables used to create the datalog files in the Embedded Microcontroller K64F MCU, in Kinetis Design Studio…

Sensor Fusion Domain

```
pit=thisSV_9DOF_GBY_KALMAN.fThePl;     // pitch
rol=thisSV_9DOF_GBY_KALMAN.fPhiPl;     // roll
yaw=thisSV_9DOF_GBY_KALMAN.fPsiPl;     // yaw
hdg=thisSV_9DOF_GBY_KALMAN.fRhoPl;     // heading
lax=thisSV_9DOF_GBY_KALMAN.faSePl[0]; // linXA
lay=thisSV_9DOF_GBY_KALMAN.faSePl[1]; // linYA
laz=thisSV_9DOF_GBY_KALMAN.faSePl[2]; // linZA
icx=thisMag.fBc[0];                    // calXM
icy=thisMag.fBc[1];                    // calYM
icz=thisMag.fBc[2];                    // calZM
```

…and they happen to be the **EXACT SAME** variables used in the JAVA GUI on the hands-on labs.

Variables available to you when you write your Algorithms in the Java GUI…

Algorithms.java

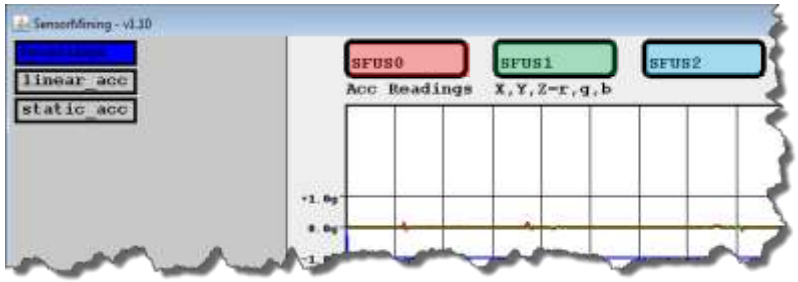
```
11
12 private final int fftn=1024;
13 private int   fftsec;
14 private float fft[];
15
16 private float pit,rol,yaw;
17 private float hdg;
18 private float lax,lay,laz;
19 private float icx,icy,icz;
```

Let's jump back into the Java GUI algorithms…assigning algorithms to GUI buttons names in Sensor Fusion Domain

- In the labs, when you write your Algorithms, you will get to assign the button names for up to ten buttons (ten algorithms) for each of the labs in the **Sensor Fusion Domain**, in the examples below:

```
// LG3 Algorithms (SENSOR FUSION)
//-------------------------------------------------------------
private int LG3algo0()
{
int res=0;
if ((hdg>355f)||(hdg<5f)) res=2;
return res;
}
//-------------------------------------------------------------
private int LG3algo1()
{
int res=0;
     if ((hdg>355f)||(hdg<  5f)) res=2;
else if ((hdg>170f)&&(hdg<190f)) res=1;
return res;
}
//-------------------------------------------------------------
private int LG3algo2()
{
int res=0;
float h;
h=hdg+22.5f; if (h>360f) h=h-360f;
res=(int)(h/45f)+1;
return res;
}
```

Sensor Fusion Domain Screen



LG1algo0() is given the button name "SFUS0"
LG1algo1() is given the button name "SFUS1"
LG1algo2() is given the button name "SFUS2"

Let's compare the Java GUI to Microcontroller Embedded Code for the Sensor Fusion Domain

Embedded Microcontroller Code from the MK64F MCU. This is the code used to try your algorithm in the Java GUI on **REAL** hardware taken from the Sensor Fusion Code.

```
void UserMediumFrequencyTaskRun(void)
{
switch (runstate)
    {
    case 0: // reset values
        seconds=0; RGB(0,0,1);
        Algorithm_init();
        runstate=1;
        break;
    case 1: // wait for calibration (give up after 20 seconds)
        if ((seconds>=20*25)||(thisMagCal.fFitErrorpc<5.0f)) {RGB(0,0,0); seconds=0; runstate=2;}
        break;
    case 2: // go Red for 5 seconds
        pit=thisSV_9DOF_GBY_KALMAN.fThePl;    // pitch
        rol=thisSV_9DOF_GBY_KALMAN.fPhiPl;    // roll
        yaw=thisSV_9DOF_GBY_KALMAN.fPsiPl;    // yaw
        hdg=thisSV_9DOF_GBY_KALMAN.fRhoPl;    // heading
        lax=thisSV_9DOF_GBY_KALMAN.faSePl[0]; // linXA
        lay=thisSV_9DOF_GBY_KALMAN.faSePl[1]; // linYA
        laz=thisSV_9DOF_GBY_KALMAN.faSePl[2]; // linZA
      icx=thisMag.fBc[0];                     // calXM
        icy=thisMag.fBc[1];                   // calYM
        icz=thisMag.fBc[2];                   // calZM
        LEDcolor=Algorithm_run();
      switch (LEDcolor)
        {
        case 0: RGB(0,0,0); break;  // LED off
        case 1: RGB(1,0,0); break;  // LED=red
        case 2: RGB(0,1,0); break;  // LED=green
        case 3: RGB(0,0,1); break;  // LED=blue
        case 4: RGB(1,1,0); break;  // LED=yellow
        case 5: RGB(1,0,1); break;  // LED=magenta
        case 6: RGB(0,1,1); break;  // LED=cyan
        case 7: RGB(1,1,1); break;  // LED=white
        case 8: RGB(0,0,0); break;  // LED=off
        }
        break;
    }
}
```

25Hz Loop

Captured Datalog Raw file from Sensor Fusion stations

```
735  02DE-[FF00008A2094FE0A013A0068001200DB0035....3F8E241E3E12207F433263E433263
736  02DF-[FF08007C2096FE06014000760012000C0039....3F8C0EF83E1D931F4332494C43324940
737  02E0-[FF0200832094FE0A0144007400C000A0039....3F8CAE873E33EE3B433229B9433229B9
738  02E1-[FF0700822094FE0B013F0074000E000A003B....3F8D42D73E42003343320C4243320C42
739  02E2-[FF0100812094FE0D013E0076000A000D0039....3F8C08B53E4043884331F3E34331F3E3
740  02E3-[FF0300822094FE12013F0070000C000F0038....3F8B34D13E3B81FC4331DC124331DC12
741  02E4-[FF09007E20A0FE10013E0070001000010037....3F86942A3E42B5A04331CD934331CD93
742  02E5-[FF080084209AFE0A0139006A001000060039....3F8671F83E3A80274331BE134331BE13
743  02E6-[FF0600852099FE04013D0069001100A0032....3F847A483E540D5D43319EA543319EA5
744  02E7-[FF03008209EFE0D0142006E0013000A0037....3F81DA6A3E47154243331902E4331902
745  02E8-[FF0000882094FE12013C0067001000070039....3F7EC0D53E472C2D4331D7D843317D7D
746  02E9-[FF04007F209EFE08013A006E000F000E0038....3F7E53DD3E415B9443316570433165703
747  02EA-[FF0400822094FE0B0138007200OF00060040....3F76698A3E4D44F84331494943314949
748  02EB-[FF06007F209EFE0A0139006600130000C0038....3F6D7AEB3E4460C14331366A4331366A
749  02EC-[FF0200862094FE0C0140006500OF0006003B....3F6DCA403E5A3D1E433116C1433116C1
750  02ED-[FF07008209DFE08014200GF000F0006003A....3F6A26C53E76E9B84330F5194330F519
```

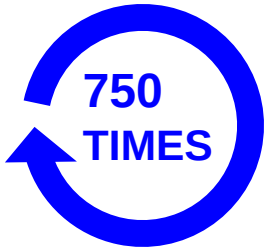
The *UserMediumFrequencytaskRun(void) function* runs at 25Hz for 30 seconds. So there will be 25 samples/sec x 30 sec = 750 entries of raw data. Well, that is EXACTLY what the data looks like when we look at the raw data log file. 750 ENTRIES!

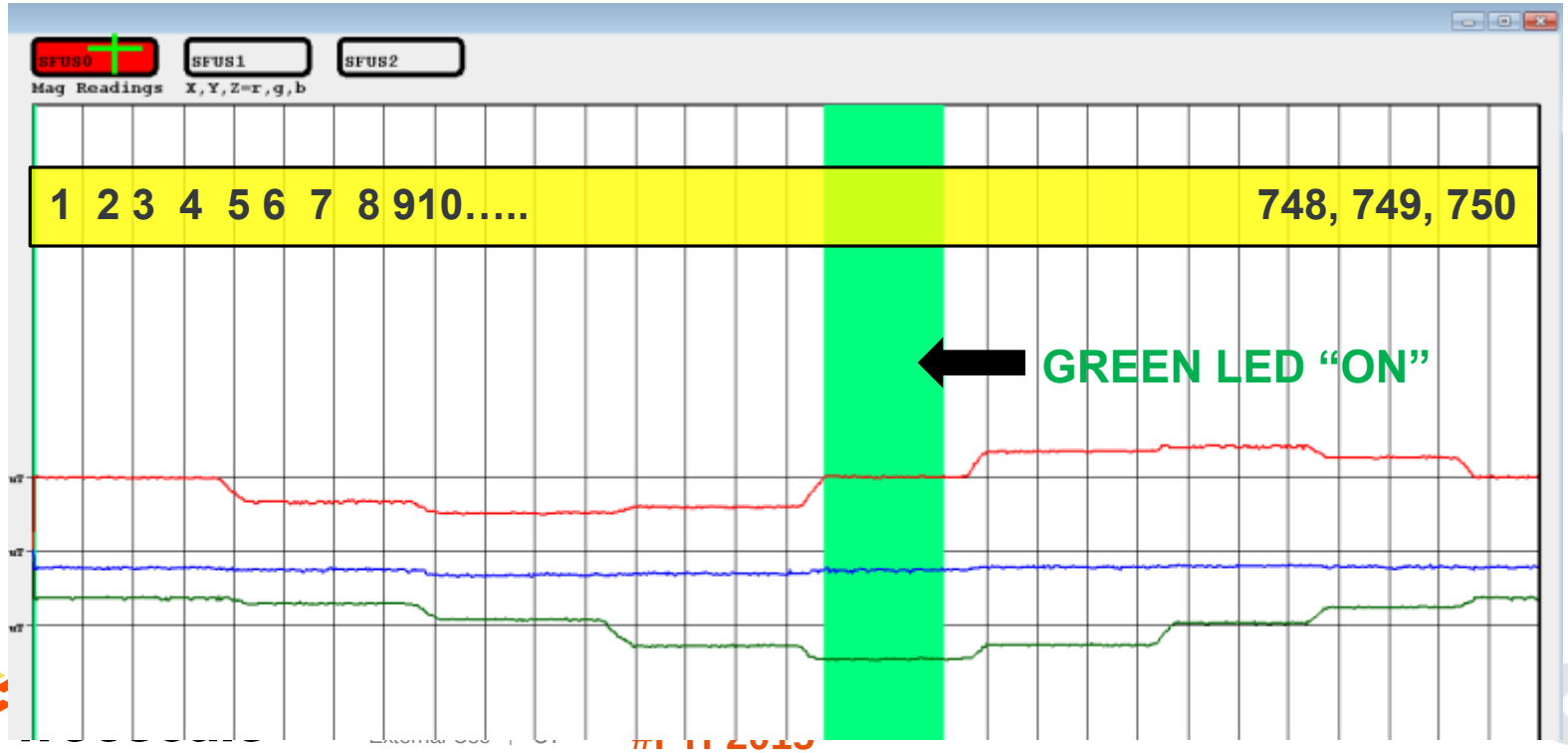
How does the Java Algorithm run on the Sensor Mining GUI in the Sensor Fusion Domain?

- Here is the most important part of understanding the Sensor Mining GUI…

ALGORITHMS WILL EXECUTE 750 times when the button is pressed!

```
// LG3 Algorithms (SENSOR FUSION)
// ======================================
private int LG3algo0()
{
int res=0;
if ((hdg>355f)||(hdg<5f)) res=2;
return res;
}
```





…and (optionally) load onto your FRDM-K64F and test it!

…after you have proved you algorithm in the "Sensor Mining Java GUI"…you can load you code into the REAL hardware…

…and see your "Sensor Mining JAVA" algorithm run on REAL hardware with the LEDs lighting up!

Add Videos of Tuning Fork

And Sensor Fusion compass

Sensor Strengths & Weaknesses

Sensor	Strengths	Weaknesses
Accelerometer	• Inexpensive • Extremely low power • Very linear • Very low noise	• Measures the sum of gravity and acceleration. We need them separate.
Magnetometer	• The only sensor that can orient itself with regard to “North” • Insensitive to linear acceleration • Great a measure absolute position	• Subject to magnetic interference • Not “spatially constant”
Gyro	• Relatively independent of linear acceleration • Can be used to “gyro-compensate” the magnetometer	• Power hog • Long startup time • Zero rate offset drifts over time
Pressure Sensor	• The only stand-alone sensor that can give an indication of altitude	• Not well understood • A “relative” measurement • Subject to many interferences and environmental factors

Q&A

- Did you understand the class objectives?
- Did you write your own algorithms?
- Did you understand the algorithms we provided, and our methodology?

Free Tools We Used

Tools work in Windows, Mac, and Linux!

- Eclipse IDE for Java Developers (we used Luna)
  - https://eclipse.org/downloads/

- Java JDK (we used Java JDK8u45)
  - http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

www.Freescale.com