# DSP Final Project

- Your team has been hired to build a breathing rate detection system to detect potential acute respiratory infection (pneumonia) in children aged 11 months to 5 years

# System Description

- Build a breathing rate monitor to warn of a potential acute respiratory problem (pneumonia) in a child age 11 months to 5 years.
- In children, a breathing rate of greater than 40 breaths per minute can indicate pneumonia.
- Fewer than 12 breaths per minute can also indicate an abnormal condition

# System Requirements

- Monitor shall detect if the breathing rate is greater than 40 breaths per minute
- Monitor shall detect a breathing rate below 12 breaths per minute
  - May indicate that the sensor is disconnected, or other abnormal condition is occurring
- Either condition shall be detected and alert sounded within 2 minutes of its occurrence

# System Requirements

- A warning shall be sounded for a breathing rate greater than 40 breaths per minute
- A warning shall be sounded for a breathing rate less than 12 breaths per minute
- The warning for low breathing rate shall be different from that of the warning for high breathing rate

# Final Project Approach

- You have 3 weeks to complete the project
- Each week there are suggested activities to work on for your project.
  - You may implement whatever solution your group decides
- You will have to invest significant time outside of lab to complete the final project
- Your group must work together to get a passing grade

# Weekly Suggested Activities

- Week 01
  - Build and test filter banks
  - Build and test running statistics
- Week 02
  - Write detection logic
  - Write code to sound the warning
- Week 03
  - Complete integration of all hardware and software
  - Test, Create Video, Write report

# Weekly Suggested Activities

- There are no reports due until the final project report
  - Due 12/9 -- No Late Reports/Videos
- If you finish the activity for the week, keep moving forward
- Make sure that you have more than one person with a working set of hardware
- Start with the base code and build upon that each week

# Project Suggestions

- This project will require a significant amount of work
  - Create a plan
- Every team member must participate
- Determine who is going to do what
  - Code sections
  - Test sections
  - Prepare video
  - Work on report
- Be accountable for your tasks
- Work together as a team!!!

# What's the Problem?

- Pediatric Pneumonia

https://www.youtube.com/watch?v=v-EclaR97y8

# Some Ideas

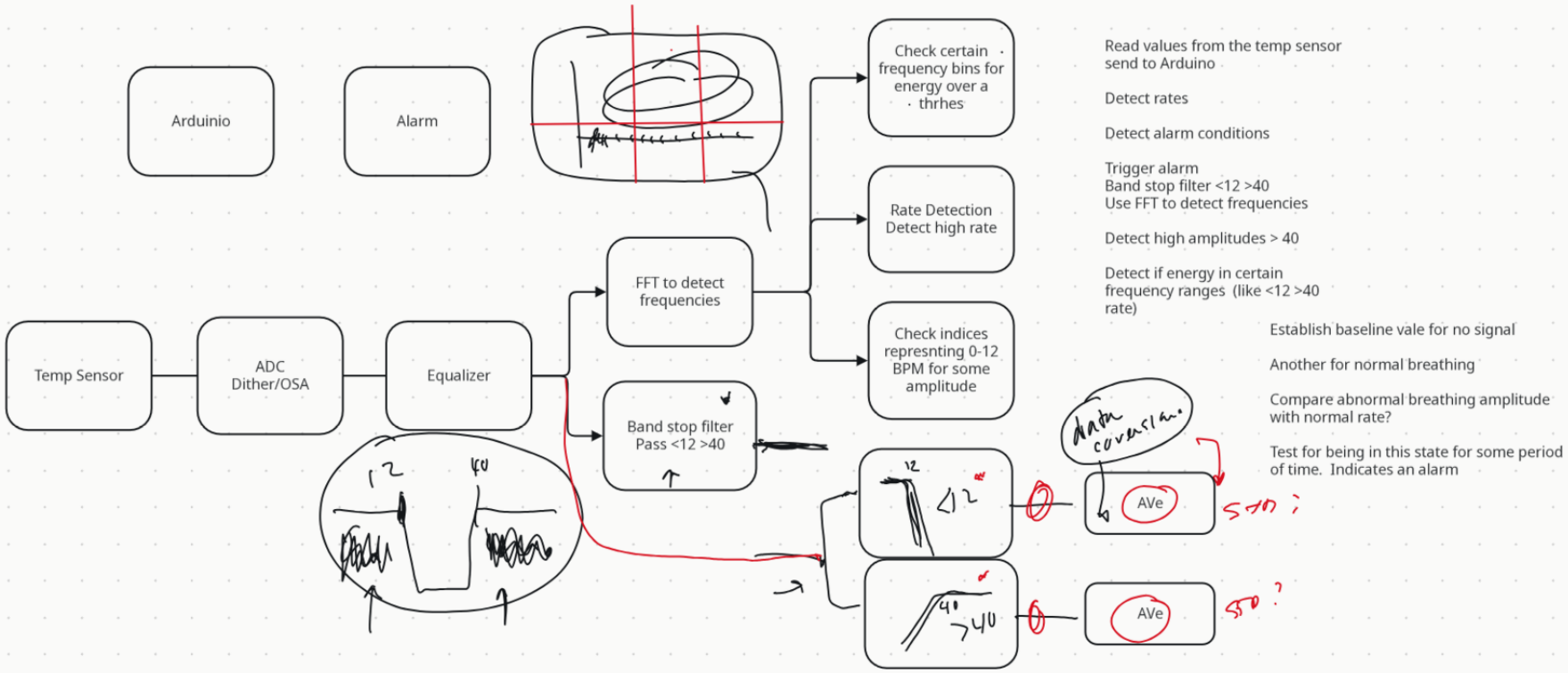
- These are notes on the discussion. There is no assessment on the feasibility or complexity of these approaches. Just notes

- Signal Conditioning
  - Use dithering and oversampling and averaging to get a good reading from the ADC
  - Follow that with the equalizer to equalize the temperature sensor frequency response
- Also need an alarm function

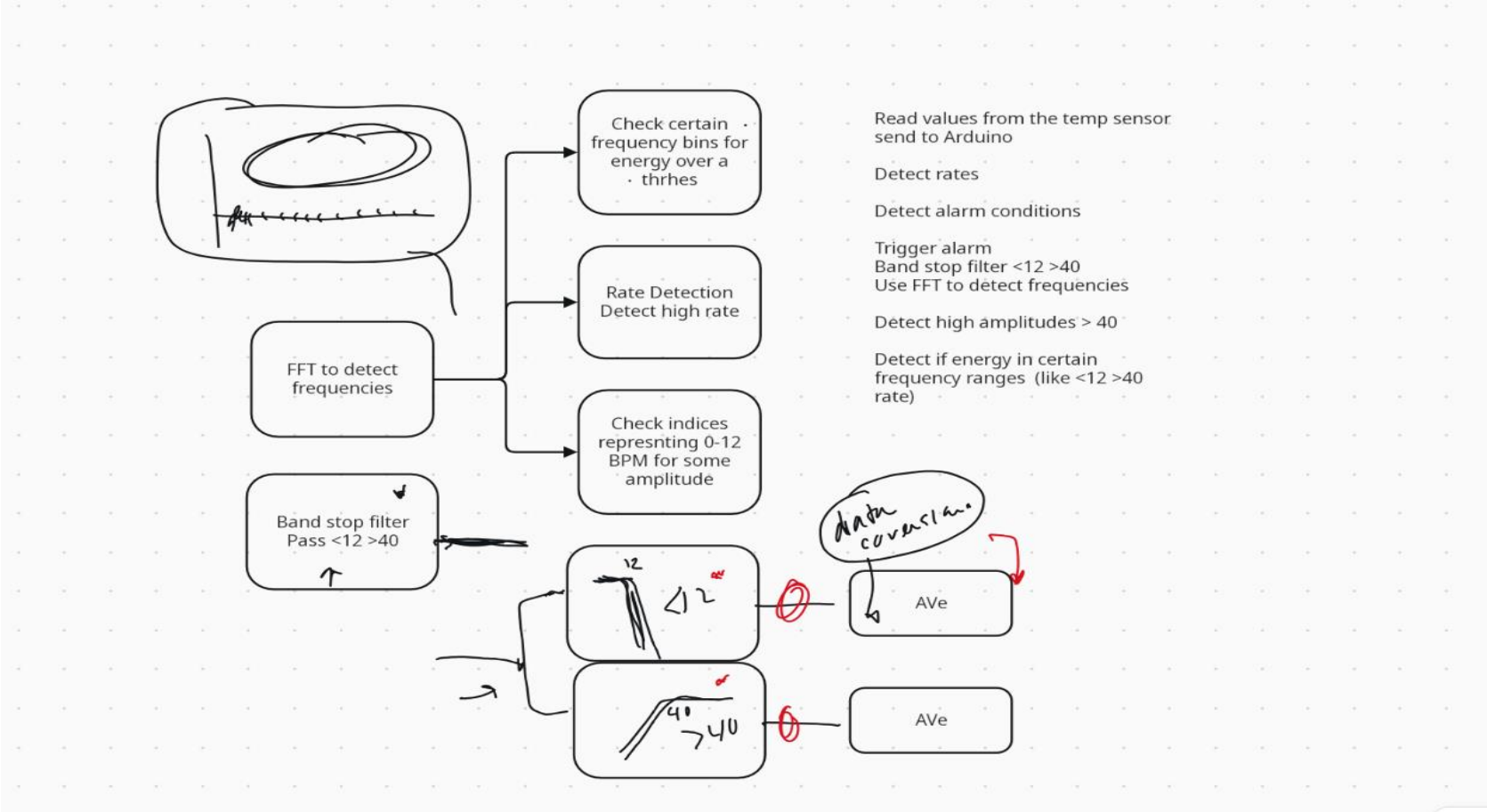
# Some Ideas

- Frequency Detection/Separation
  - Use a filter or filters to separate frequency ranges of signals
    - Band stop or Lowpass and High pass
    - Average the values out of the filter to determine amplitude
    - Use the standard deviation after the filter to determine amplitude
  - Use an FFT to look at the signal in the frequency domain. Determine levels of signals in different bin ranges
    - Check energy within bins that represent the ranges of frequency of interest (low freq range, mid freq range, high freq range)
    - Compare values from the ranges to determine the breathing rate
  - After the Equalizer differentiate the signal to determine the frequency value
    - Find the standard deviation of the output
    - Use a moving average filter to find the range

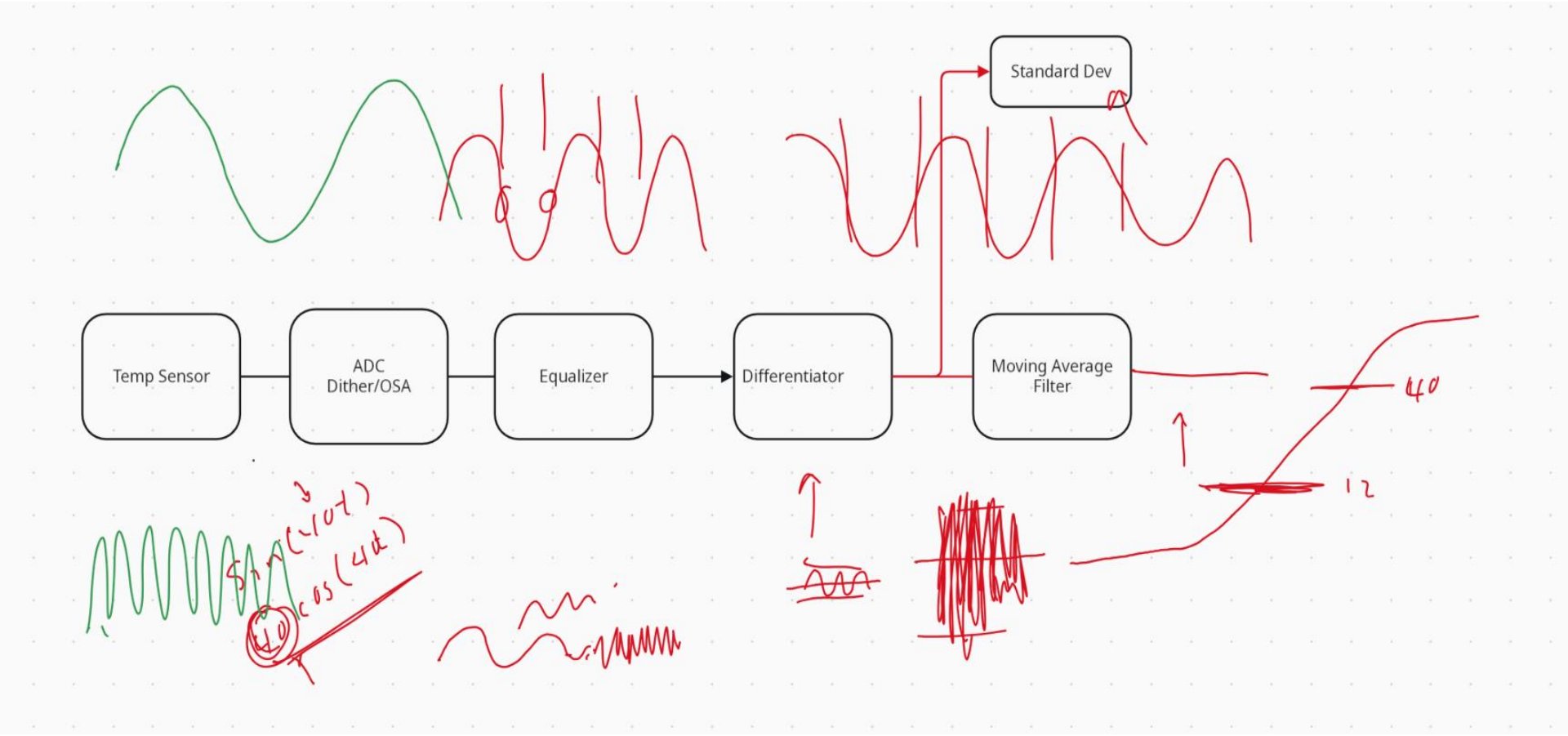
# Brainstorming
# Pre-processing, FFT, Filter Banks



# More detail on FFT and Filter Approach

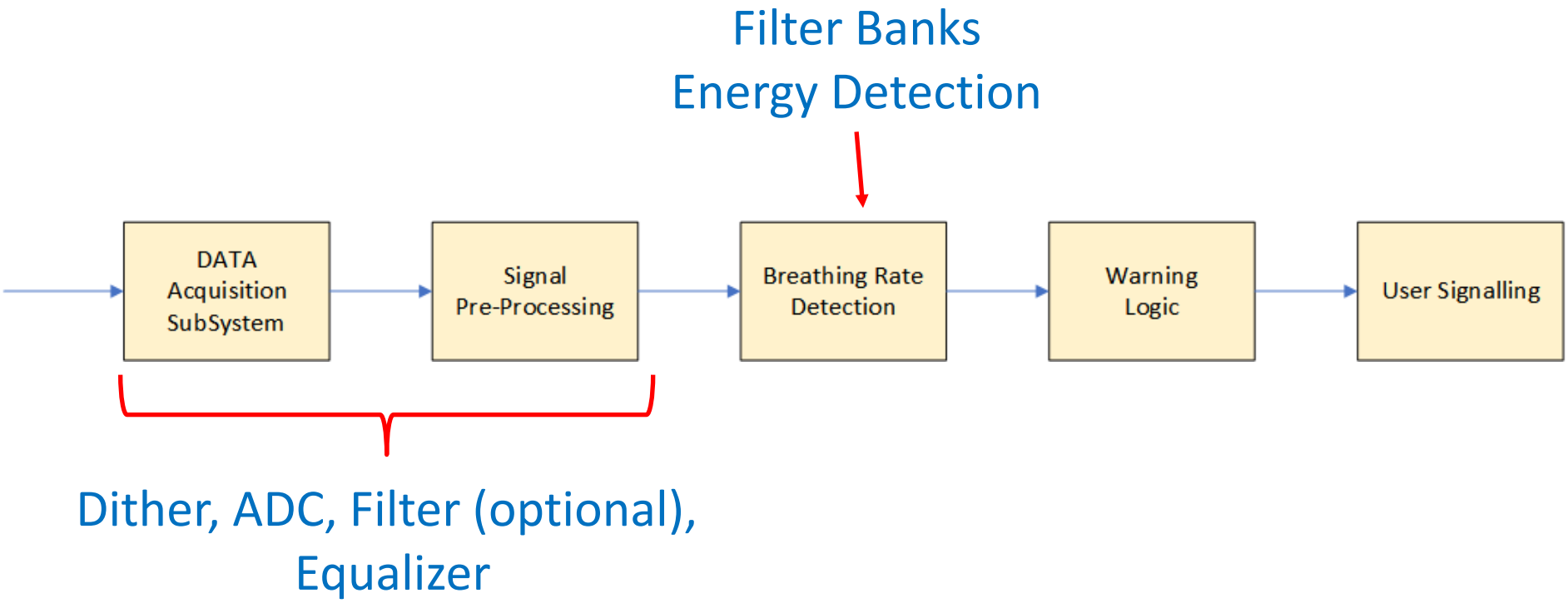


# Differentiator
# Frequency Detector approach

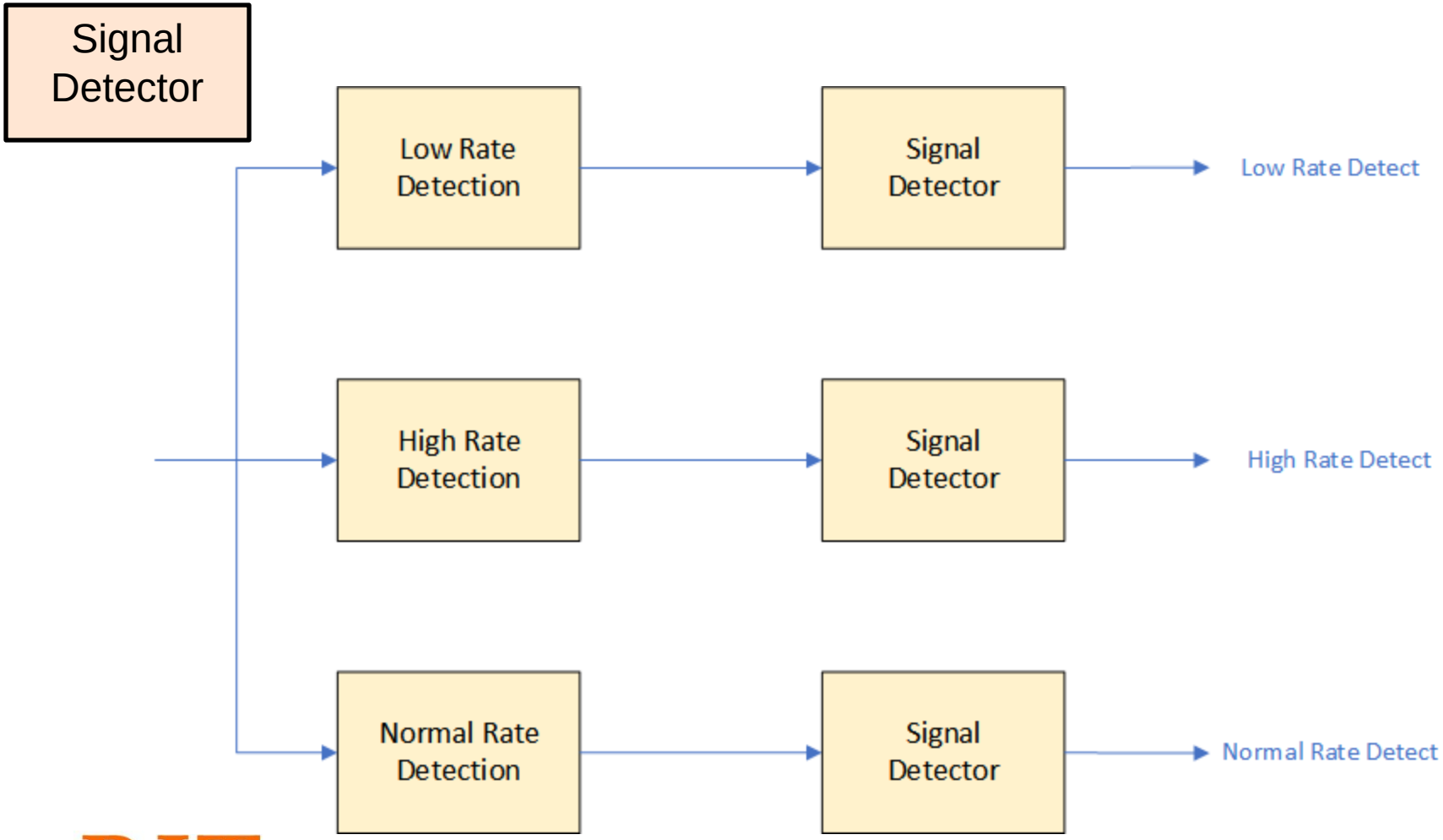


# 10,000 Foot Level

- At the very highest level a block diagram for the system might look like this



# Filter Banks (General Approach)



# Digital Signal Processing

## Final Project Week 01
## Filter Bank Design

# Week 01 Objectives

- Start to put together pieces for the final project
- Implement a bank of filters to provide frequency separation
- Run filters in parallel

# Week 01 Objectives

- Implement a resettable statistics function
- Run statistics function in parallel

# A More General Way to Write to the Serial Port

- In many of the earlier lab the function ‘displayData’ was called to write to the serial port
  - It is specific to the data that is to be written
- In the project you may want to write many different types of data to the serial port
  - Need a more general way to write the data
  - Avoids re-writing ‘displayData’ for each change.
  - ‘WriteToSerial’ is the new function

# A More General Way to Write to the Serial Port

- Instead of changing 'displayData' function just write different value to the variable 'printArray'

Set values in 'printArray' to the value to send to the port

'printArray[0] is always loopTick (sample Number). Don't' change!

```
// To print data to the serial port, use the WriteToSerial function.
//
//  This is a generic way to print out variable number of values
//
//  There are two input arguments to the function:
//  printArray -- An array of values that are to be printed starting with the first column
//  numValues -- An integer indicating the number of values in the array.

  printArray[0] = loopTick;  //  The sample number -- always print this
  printArray[1] = xv;        //  Column 2

//   printArray[2] = yLF;       //  Column 3
//   printArray[3] = yMF;       //  Column 4, etc...
//   printArray[4] = yHF;
//   printArray[5] = stdLF;
//   printArray[6] = stdMF;
//   printArray[7] = stdHF;
//   printArray[8] = float(alarmCode);

  numValues = 2;  // The number of columns to be sent to the serial monitor (or MATLAB)

WriteToSerial( numValues, printArray );  //  Write to the serial monitor (or MATLAB)
```

# A More General Way to Write to the Serial Port

- Instead of changing ‘displayData’ function just write different value to the variable ‘printArray’

Uncomment other array values as needed

Set ‘numValue’ to the number of variables passed

( Always include printArray[0] )

```
// To print data to the serial port, use the WriteToSerial function.
//
//  This is a generic way to print out variable number of values
//
//  There are two input arguments to the function:
//  printArray -- An array of values that are to be printed starting with the first column
//  numValues -- An integer indicating the number of values in the array.

  printArray[0] = loopTick;  //  The sample number -- always print this
  printArray[1] = xv;        //  Column 2

//  printArray[2] = yLF;       //  Column 3
//  printArray[3] = yMF;       //  Column 4,
//  printArray[4] = yHF;
//  printArray[5] = stdLF;
//  printArray[6] = stdMF;
//  printArray[7] = stdHF;
//  printArray[8] = float(alarmCode);

  numValues = 2;  // The number of columns to be sent to the serial monitor (or MATLAB)

WriteToSerial( numValues, printArray );  //  Write to the serial monitor (or MATLAB)
```

Call ‘WriteToSerial’ with ‘numValues’ and ‘printArray’ as arguments

# Preventing Serial Port Overload

- The “CaptureArduinoData.m” routine includes a parameter “GraphDelay”
  - Prevents serial port overload from occurring
  - Add “GraphDelay”, N to the function call

```
>> data = CaptureArduinoData('ComPort',3,'BaudRate',115200,'NumActivePlots',4,'GraphDelay',100);
```

  - N=100 samples works well

# Week 01 Steps

- Design and implement
  - Low frequency filter
  - Mid range filter
  - High range filter
  - Run them in parallel
  - Add running statistics running in parallel
  - Measure execution time