

## MEET THE TEAM

Kyle Buchanan  
Theodore Pham  
James Cheng

## Motivation

- Technically challenging
- Good way to create backup notes
- Could be utilized by groups such as SSOS
- Lots of room to expand or narrow scope

## Functionality

- Recognition of shapes, symbols, and numbers on a flat surface
- Wireless communication of written character from pen to board
- Incorporates an accelerometer to measure the acceleration of the pen - Built in non-tiltable pen caps for visibility of written work
- Displays user written input on a LCD screen
- Uses flash memory to store the character templates
- User click to activate writing sensor
- Power button to turn the pen on/off

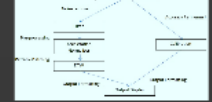
## Pattern Recognition

- Selected Algorithm is Dynamic Time Warping
- Finds optimal matches between time series
- Used in other pattern matching applications
- Uses dynamic programming to find ideal paths

## Pattern Recognition



## Design: Software



## Design Calculations

- Data Filtering: 44 FLOPS
- DTW algorithm:
  - time cost:  $O(n^2)$
  - space cost:  $O(n)$
- Battery Runtime on 8 h wake/ 4 h sleep cycles
  - Coin Cell Battery: ~3.00 h
  - Rechargeable AA: ~29.40 h
- I/O rates: 36 bits/interval
- Xbee modules: 625 bits per 2.5 ms
- Flash memory required per character: ~3.5 kB
- Onboard Flash: 4 MB == 1165 patterns

## Challenges Thus Far

- The Bluetooth module could not act as a Master to poll data from the accelerometer without a microcontroller so we decided to use wireless Xbee module instead.
- Trying to fit all components into a reasonable sized pen body.
- Accurately prediction of accelerometer output data

## Optional Features

- Extra button to change pen functionality to perform basic arithmetic
- Option to allow pen to be user dependent (calibrates pen according to users initial written input pattern)
- Ability to recognize alphabets (in future - handwriting)
- Design a GUI for character display on the computer
- Sensor activated without use of a button trigger or pressure sensor
- Incorporate Bluetooth and integrate this new design so that users can connect the pen to their smartphones and generate outputs to an app



## Results of Unit and Integration Testing

DTW algorithm test  
- Test input with 400 data points  
- runs in 0.172s using an i7@3.40hz

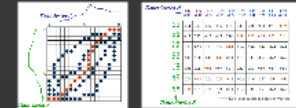
## Application Notes

Although we do not have application notes of our own yet, we are using an Xbee module from a past project on 2012.

## Design: graphical hierarchy of source code



## Design: Software



## Awesome Code

```

int main()
{
    // Initialize variables
    int x, y, z;
    // ...
}

```

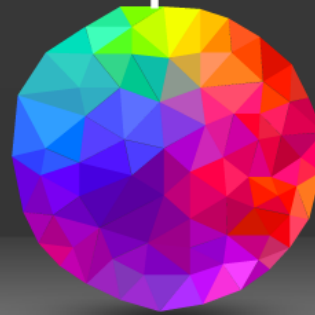
## Components

- Implementation on B2 Board
  - Xbee transceiver module
  - USB connection to computer
- Implementation on Pen Unit
  - Xbee transceiver module
  - Accelerometer
  - Power Source
  - Switches (Trigger, Power)

## Design: Hardware



# Accelerometer Pen CRITICAL DESIGN



# Accelerometer Pen

## CRITICAL DESIGN



# MEET THE TEAM

Kyle Buchanan

Theodore Pham

James Chang



# Motivation

- Technically challenging
- Good way to create backup notes
- Could be utilized by groups such as SSDS
- Lots of room to expand or narrow scope



# Functionality

- Recognition of shapes, symbols, and numbers on a flat surface
- Wireless communication of written character from pen to board
- Incorporates an accelerometer to measure the acceleration of the pen - Built in non-fillable pen cartridge for visibility of written work
- Displays user written input on a LCD screen
- Uses flash memory to store the character templates
- User click to activate writing sensor
- Power button to turn the pen on/off



# Pattern Recognition

- Selected Algorithm is Dynamic Time Warping
  - Finds optimal matches between time series
  - Used in other pattern matching applications
  - Uses dynamic programming to find ideal paths

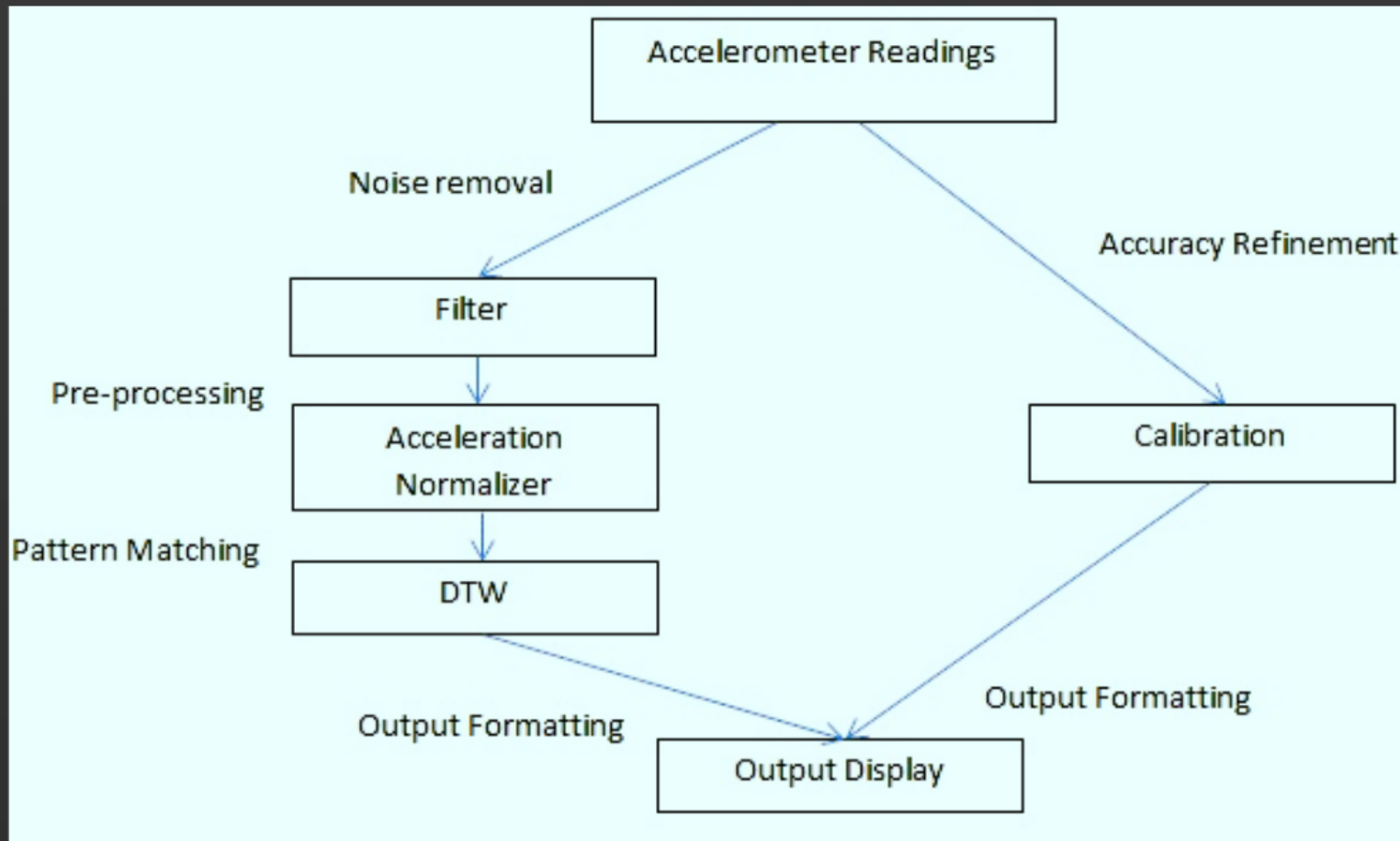




# Pattern Recognition



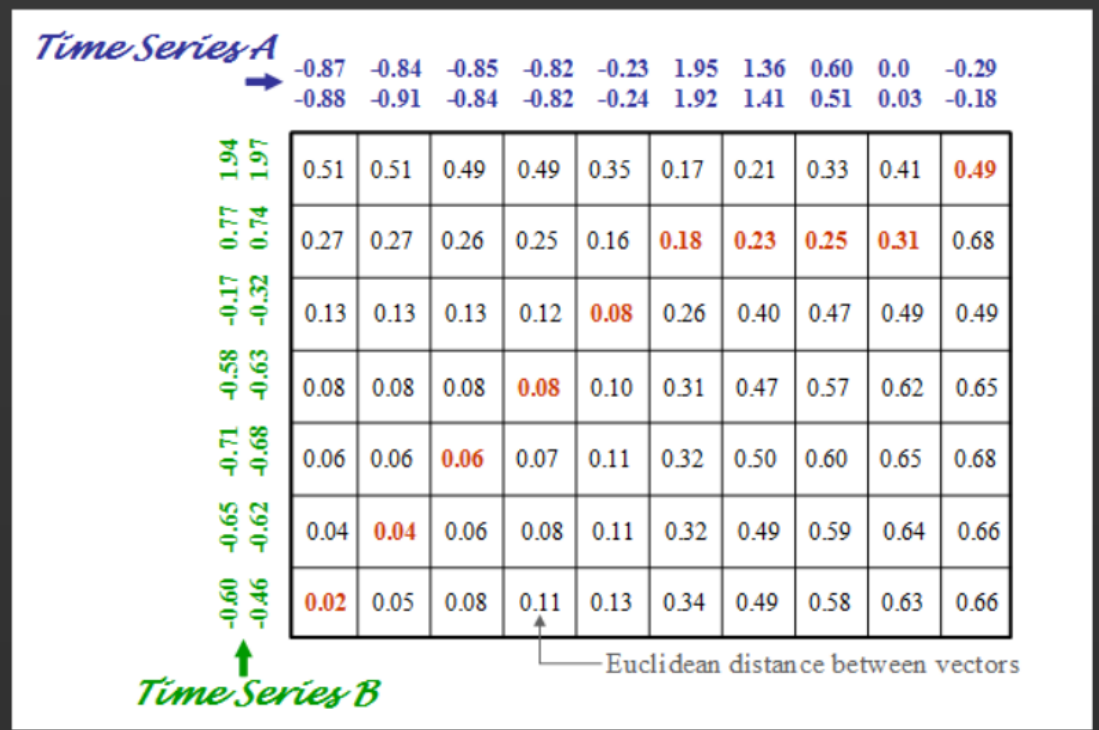
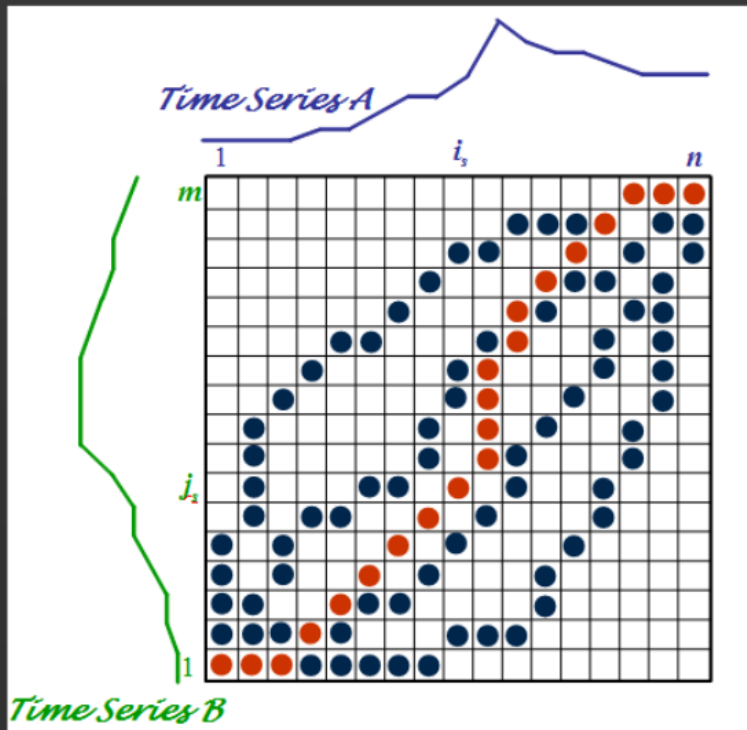

# Design: Software





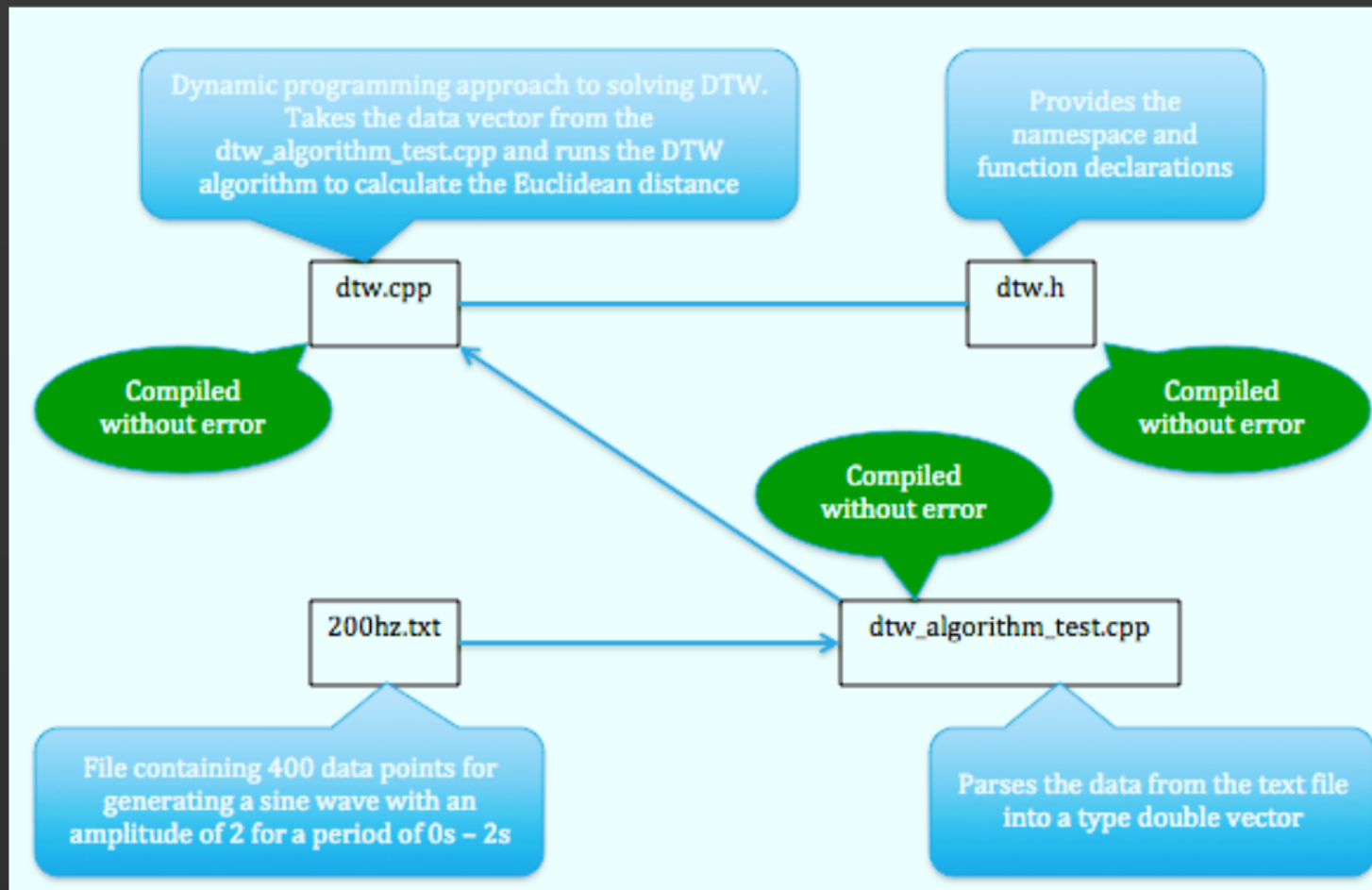


# Design: Software





# Design: graphical hierarchy of source code





# Awesome Code

```
#include "dtw.h"
#include <iostream>

namespace DTW {

    double dist(double x, double y) {
        return sqrt(pow((x - y), 2));
    }

    //dynamic programming approach
    double dtw(const std::vector<double>& t1, const std::vector<double>& t2) {
        int m = t1.size();
        int n = t2.size();

        // create cost matrix
        double cost[m][n];
        cost[0][0] = dist(t1[0], t2[0]);

        // calculate first row
        for(int i = 1; i < m; i++)
            cost[i][0] = cost[i-1][0] + dist(t1[i], t2[0]);

        // calculate first column
        for(int j = 1; j < n; j++)
            cost[0][j] = cost[0][j-1] + dist(t1[0], t2[j]);

        // fill matrix
        for(int i = 1; i < m; i++)
            for(int j = 1; j < n; j++)
                cost[i][j] = std::min(cost[i-1][j], std::min(cost[i][j-1], cost[i-1][j-1]))
                    + dist(t1[i], t2[j]);

        return cost[m-1][n-1];
    }
}
```

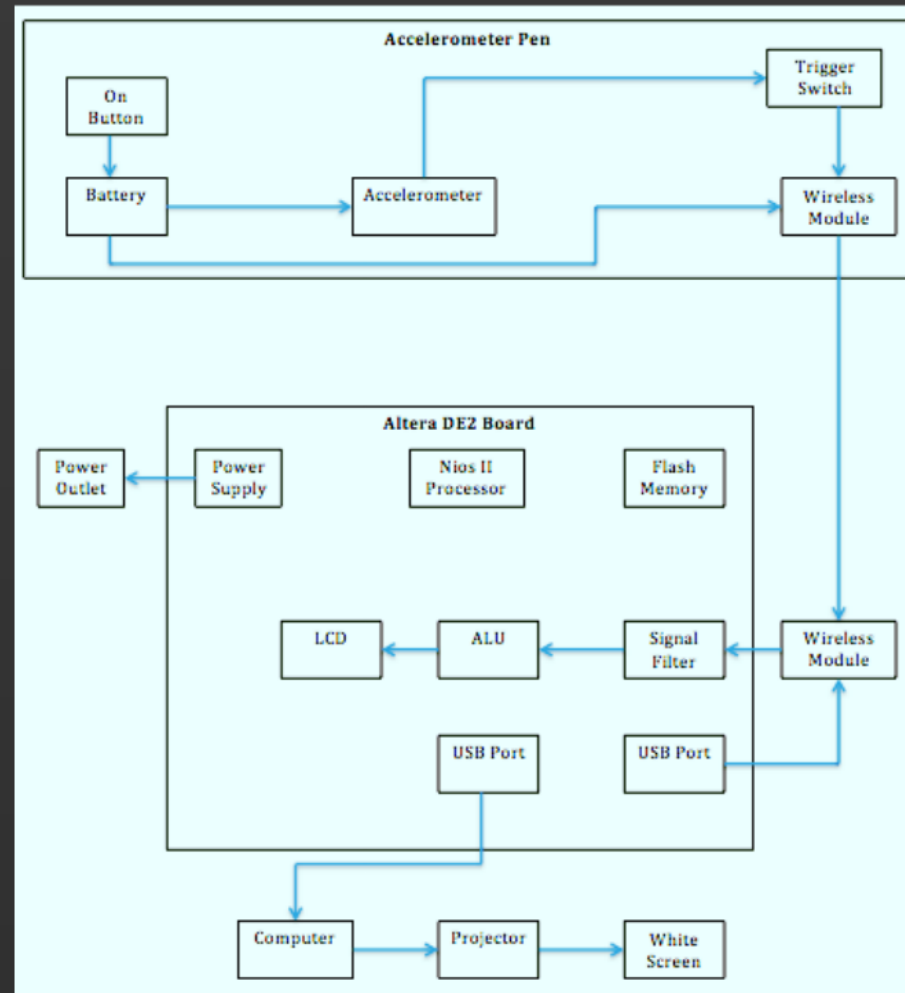


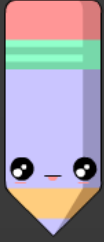
# Components

- Implementation on DE2 Board:
  - Xbee transceiver module
  - USB connection to computer
- Implementation on Pen Unit:
  - XBee transceiver module
  - Accelerometer
  - Power Source
  - Switches (Trigger, Power)



# Design: Hardware





# Design Calculations

- Data Filtering: 44 FLOPS
- DTW algorithm:
  - time cost:  $O(n^2)$
  - space cost:  $O(n)$
- Battery Runtime on 8 h wake/ 4 h sleep cycles
  - Coin Cell Battery: ~3.00 h
  - Rechargeable AA: ~29.40 h
- I/O rates: 36 bits/interval
- Xbee modules: 625 bits per 2.5 ms
- Flash memory required per character: ~3.5 kB
- Onboard Flash: 4 MB  $\approx$  1165 patterns



# Optional Features

- Extra button to change pen functionality to perform basic arithmetic
- Option to allow pen to be user dependent (calibrates pen according to users initial written input pattern)
- Ability to recognize alphabets (in future - handwriting)
- Design a GUI for character display on the computer
- Sensor activated without use of a button trigger or pressure sensor
- Incorporate Bluetooth and integrate this new design so that users can connect the pen to their smartphones and generate outputs to an app



# Results of Unit and Integration Testing

## DTW algorithm test

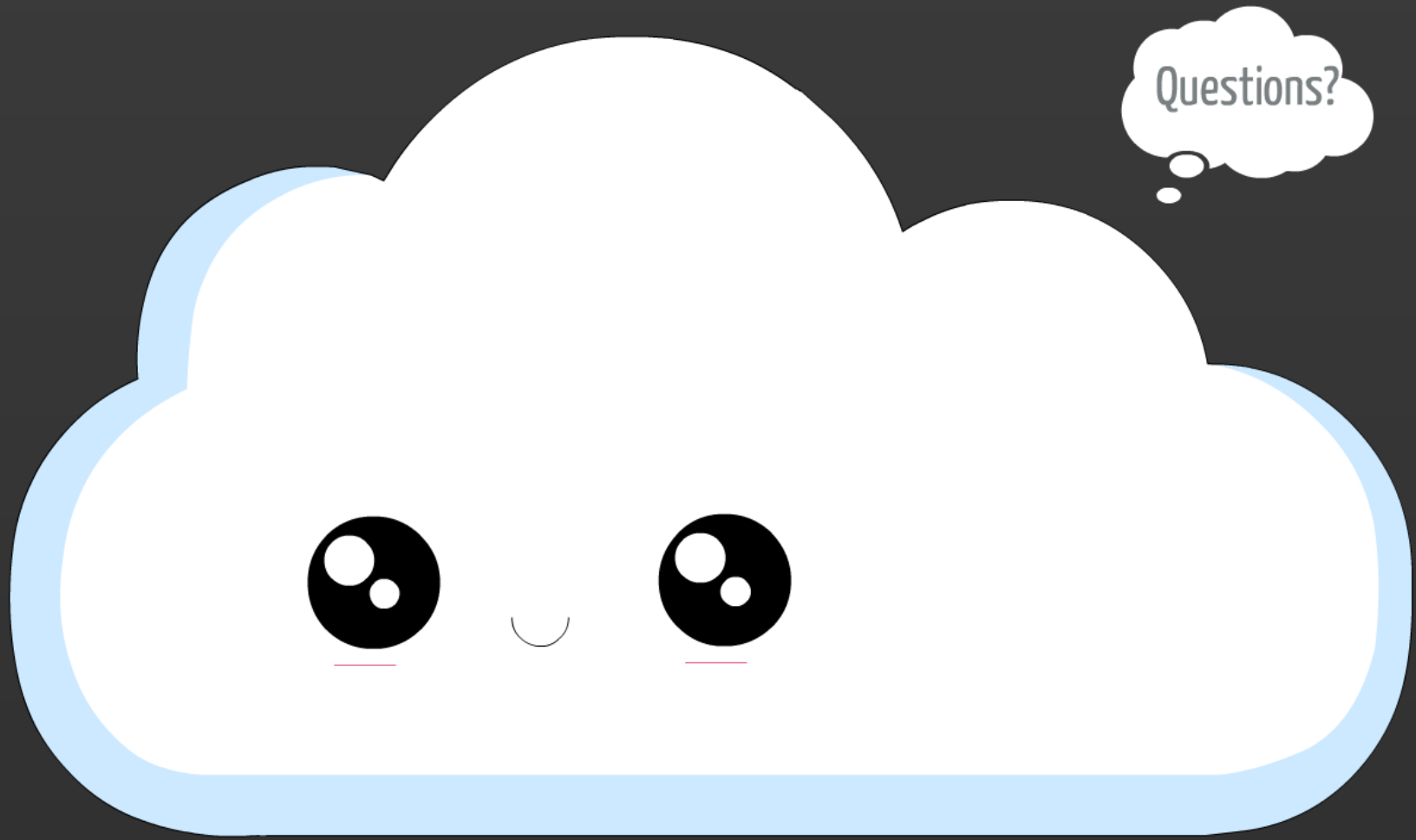
- Test input with 400 data points
- runs in 0.171s using an i7@3.4GHz





# Challenges Thus Far

- The Bluetooth module could not act as a Master to poll data from the accelerometer without a microcontroller so we decided to use wireless Xbee module instead.
- Trying to fit all components into a reasonable sized pen body.
- Accurately prediction of accelerometer output data



Questions?

## MEET THE TEAM

Kyle Buchanan  
 Theodore Pham  
 James Chang

## Motivation

- Technically challenging
- Good way to create backup notes
- Could be utilized by groups such as SSOS
- Lots of room to expand or narrow scope

## Functionality

- Recognition of shapes, symbols, and numbers on a flat surface
- Wireless communication of written character from pen to board
- Incorporates an accelerometer to measure the acceleration of the pen - built in non-fillable pen cartridge for visibility of written work
- Displays user written input on a LCD screen
- Uses flash memory to store the character templates
- User click to activate writing sensor
- Power button to turn the pen on/off

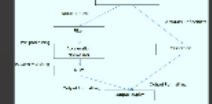
## Pattern Recognition

- Selected Algorithm is Dynamic Time Warping
- Finds optimal matches between time series
- Used in other pattern matching applications
- Uses dynamic programming to find ideal paths

## Pattern Recognition



## Design: Software



## Design Calculations

- Data Filtering: 44 FLOPS
- DTW algorithm:
  - time cost:  $O(n^2)$
  - space cost:  $O(n)$
- Battery Runtime on 8 h wake/ 4 h sleep cycles
  - Coin Cell Battery: ~3.00 h
  - Rechargeable AA: ~29.40 h
- I/O rates: 36 bits/interval
- Xbee modules: 625 bits per 2.5 ms
- Flash memory required per character: ~3.5 kB
- Onboard Flash: 4 MB ~ 1165 patterns

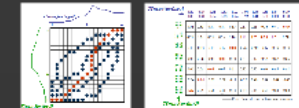
## Challenges Thus Far

- The Bluetooth module could not act as a Master to pull data from the accelerometer without a microcontroller so we decided to use wireless Xbee module instead.
- Trying to fit all components into a reasonable sized pen body.
- Accurately prediction of accelerometer output data

## Design: graphical hierarchy of source code



## Design: Software



## Optional Features

- Extra button to change pen functionality to perform basic arithmetic
- Option to allow pen to be user dependent (calibrates pen according to users initial written input pattern)
- Ability to recognize alphabets (in future - handwriting)
- Design a GUI for character display on the computer
- Sensor activated method: use of a button trigger or pressure sensor
- Integrate Bluetooth to integrate this new design so that users can connect the pen to their smartphones and generate outputs to an app

## Awesome Code

```

#include <Arduino.h>
#include <Wire.h>
#include <I2Cdev.h>
#include <MPU6050.h>
#include <Adafruit_BMP085.h>
#include <Adafruit_Sensor.h>
#include <SPI.h>
#include <SD.h>
#include <EEPROM.h>
#include <LiquidCrystal_I2C.h>
#include <LiquidCrystal.h>
#include <Servo.h>
#include <Stepper.h>
#include <SoftwareSerial.h>
#include <Dallas14631.h>
#include <Dallas14631OneWire.h>
#include <Dallas14631OneWire1.h>
#include <Dallas14631OneWire2.h>
#include <Dallas14631OneWire3.h>
#include <Dallas14631OneWire4.h>
#include <Dallas14631OneWire5.h>
#include <Dallas14631OneWire6.h>
#include <Dallas14631OneWire7.h>
#include <Dallas14631OneWire8.h>
#include <Dallas14631OneWire9.h>
#include <Dallas14631OneWire10.h>
#include <Dallas14631OneWire11.h>
#include <Dallas14631OneWire12.h>
#include <Dallas14631OneWire13.h>
#include <Dallas14631OneWire14.h>
#include <Dallas14631OneWire15.h>
#include <Dallas14631OneWire16.h>
#include <Dallas14631OneWire17.h>
#include <Dallas14631OneWire18.h>
#include <Dallas14631OneWire19.h>
#include <Dallas14631OneWire20.h>
#include <Dallas14631OneWire21.h>
#include <Dallas14631OneWire22.h>
#include <Dallas14631OneWire23.h>
#include <Dallas14631OneWire24.h>
#include <Dallas14631OneWire25.h>
#include <Dallas14631OneWire26.h>
#include <Dallas14631OneWire27.h>
#include <Dallas14631OneWire28.h>
#include <Dallas14631OneWire29.h>
#include <Dallas14631OneWire30.h>
#include <Dallas14631OneWire31.h>
#include <Dallas14631OneWire32.h>
#include <Dallas14631OneWire33.h>
#include <Dallas14631OneWire34.h>
#include <Dallas14631OneWire35.h>
#include <Dallas14631OneWire36.h>
#include <Dallas14631OneWire37.h>
#include <Dallas14631OneWire38.h>
#include <Dallas14631OneWire39.h>
#include <Dallas14631OneWire40.h>
#include <Dallas14631OneWire41.h>
#include <Dallas14631OneWire42.h>
#include <Dallas14631OneWire43.h>
#include <Dallas14631OneWire44.h>
#include <Dallas14631OneWire45.h>
#include <Dallas14631OneWire46.h>
#include <Dallas14631OneWire47.h>
#include <Dallas14631OneWire48.h>
#include <Dallas14631OneWire49.h>
#include <Dallas14631OneWire50.h>
#include <Dallas14631OneWire51.h>
#include <Dallas14631OneWire52.h>
#include <Dallas14631OneWire53.h>
#include <Dallas14631OneWire54.h>
#include <Dallas14631OneWire55.h>
#include <Dallas14631OneWire56.h>
#include <Dallas14631OneWire57.h>
#include <Dallas14631OneWire58.h>
#include <Dallas14631OneWire59.h>
#include <Dallas14631OneWire60.h>
#include <Dallas14631OneWire61.h>
#include <Dallas14631OneWire62.h>
#include <Dallas14631OneWire63.h>
#include <Dallas14631OneWire64.h>
#include <Dallas14631OneWire65.h>
#include <Dallas14631OneWire66.h>
#include <Dallas14631OneWire67.h>
#include <Dallas14631OneWire68.h>
#include <Dallas14631OneWire69.h>
#include <Dallas14631OneWire70.h>
#include <Dallas14631OneWire71.h>
#include <Dallas14631OneWire72.h>
#include <Dallas14631OneWire73.h>
#include <Dallas14631OneWire74.h>
#include <Dallas14631OneWire75.h>
#include <Dallas14631OneWire76.h>
#include <Dallas14631OneWire77.h>
#include <Dallas14631OneWire78.h>
#include <Dallas14631OneWire79.h>
#include <Dallas14631OneWire80.h>
#include <Dallas14631OneWire81.h>
#include <Dallas14631OneWire82.h>
#include <Dallas14631OneWire83.h>
#include <Dallas14631OneWire84.h>
#include <Dallas14631OneWire85.h>
#include <Dallas14631OneWire86.h>
#include <Dallas14631OneWire87.h>
#include <Dallas14631OneWire88.h>
#include <Dallas14631OneWire89.h>
#include <Dallas14631OneWire90.h>
#include <Dallas14631OneWire91.h>
#include <Dallas14631OneWire92.h>
#include <Dallas14631OneWire93.h>
#include <Dallas14631OneWire94.h>
#include <Dallas14631OneWire95.h>
#include <Dallas14631OneWire96.h>
#include <Dallas14631OneWire97.h>
#include <Dallas14631OneWire98.h>
#include <Dallas14631OneWire99.h>

```

## Components

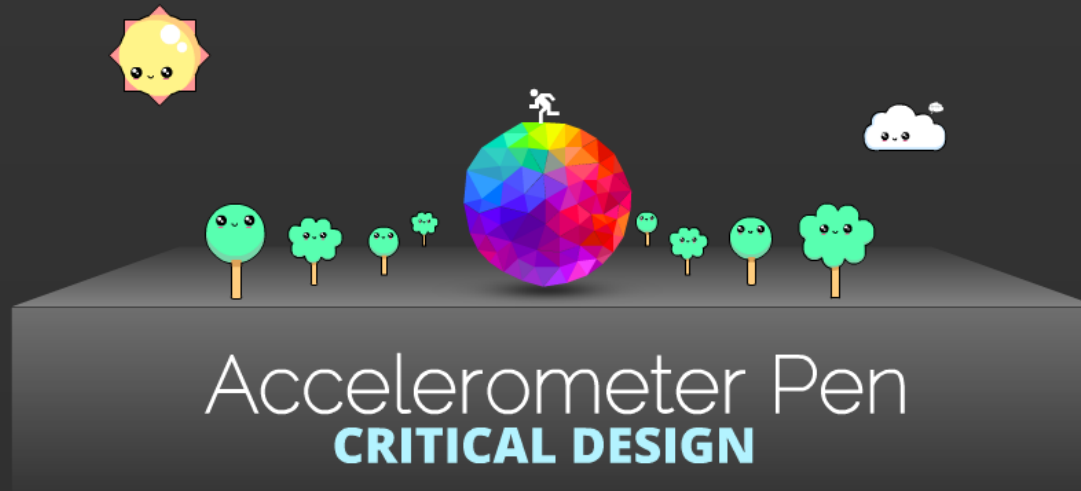
- Implementation on BE2 board:
  - Xbee transmitter module
  - USB connection to computer
- Implementation on Pen Unit:
  - Xbee transmitter module
  - Accelerometer
  - Power Source
  - Switches (Trigger, Power)

## Results of Unit and Integration Testing

DTW algorithm test  
 - Test input with 400 data points  
 - runs in 0.177s using an i7@3.4GHz

## Application Notes

- Although we do not have application notes of our own yet, we are using an Xbee module from a past project in 2012.



## Design: Hardware

