### Summary of what I have so far:

* **VI**: “ezAFM Practice 1.9”
  + I think I have a working prototype of ezAFM Automated Imaging. It is able to navigate between crosshairs and scan spots. It can take the scan and save the files to a specified folder.
* **Manual Imaging**
  + User can navigate by clicking the navigational buttons or pressing the arrow keys.
    - XY Position is recorded after each step (not very precise due to variations in step size/speed)
  + User can specify Motor Speed, Step Size, and Sample Rotation Angle
    - The sample rotation angle can be computed from the XY Position
  + User can approach, retract the tip.
* **Helper VI’s**
  + Move2Vector:
    - Takes in a Cartesian (x, y) and sends 2 commands to the XY Motor to move. Uses smaller helper VI’s to get the direction, time delays, etc.
  + GetCrosshairVector
    - User specifies a Current Crosshair and Target Crosshair and outputs the Cartesian (x, y) vector relative to each other.
  + GetScanSpot
    - User selects which spot (A0, B0, A1, B1) and outputs the Cartesian (x, y) vector relative to the crosshair center.
  + IdentifyCrosshair:
    - Takes the current ezAFM Camera image and uses “Pattern Match with Masking” VI to identify the crosshair and retrieve it’s coordinates
    - If found, then it repositions the tip
  + TakeScan:
    - Takes 3 21x21 micron scans and saves the nmi files to a specified directory.
  + BuildSequence:
    - Takes in a string list of connections and builds a string list of navigational instructions.
    - For example:
      * Move from 00 | 01 to 22 | 23 Crosshairs
      * Move to Scan Spot A1
* **Automated Imaging**
  + User selects the connections of interest to scan
    - “ezAFM Practice 1.9” incorporates Trevor’s code with using the “SimpleMap” VI (with some minor file-path issues)
    - “ezAFM Practice 1.8” uses my quick-n-dirty “SelectConnections” VI
  + The key aspect of my algorithm is that it relies on events triggering other events when they finish executing.
  + **“Automatic Repositioning”**
    - This is my version of repeatedly repositioning the tip and trying to identify the crosshair. 
    - It works in a rather crude way, it will move in a pattern resembling a square spiral.
    - Consists of 3 events: Initialization, “Search”, and “Searching”
  + **Event**:Initialize Automated Imaging:
    - Uses “BuildSequence” to set up the scanning Sequence.
    - Initializes Automatic Repositioning
  + **Event**:Automated Imaging:
    - Iterates to the next index in the Sequence to look for the substring “Scan Spot” or “Crosshairs” (ie Fine or Coarse navigation)
    - If Scan Spot, then trigger MOVE2SCANSPOT event
      * Moves to the specified scan spot, takes a scan, and moves back to the crosshair center.
    - If Crosshairs, then trigger “MOVE2CROSSHAIRS event
      * Moves to the target crosshair.
  + Within both MOVE2SCANSPOT and MOVE2CROSSHAIRS event is a call back to “Automated Imaging”, which then checks the new step in the sequence, and so on.

### Summary of what’s not finished so far:

* My helper VI’s and main VI has little to no means of error handling.
* The algorithm to find the Sample Rotation Angle from the XY Position is not reliable because the XY Position values (both my custom XY Position and the XY Motor Position) are not precise.
  + I just use a generic line tool to measure the angle of the electrodes to get an estimate. This has proven to work somewhat sufficiently.
* This VI cannot take accurate images (capture all connections) when sample rotation is more than about 4 degrees.
* This VI does not include an image pane on the front-panel to view scanned images.
* This VI’s front-panel is messy with all my helper indicators that I use to keep track of certain values (ie Sequence Index, Connection Name).
  + Moreover, I need to add some sort of user-friendly introduction for new users.