Scheduler GUI

* **Overview**
  + Just one of my tasks was to build the client UI to create schedules
  + Allows the user to create/edit events and reference audio files
  + Prevents any events from overlapping
  + Allows browsing events by date ranges
  + Allows repeating events from one range of days to another
  + Encapsulates a “schedule” as a .zip file with the schedule.xml and audio files
    - This is used to export the schedule to the box in the field
* **Demo the GUI**
  + Start the scheduler
  + Add audio
  + Create event
  + Open saved schedule
  + Copy and repeat a range of events
* **Discuss Design**
  + Implemented in C#. Leverages .NET framework whenever possible.
  + Stores each event in a ScheduledEvent object
  + Uses a BindingList<ScheduledEvent> to bind to the GridView
  + The ScheduledEvents are serialized as XML
* **Background**
  + Initially, it was my intention to have the client GUI also process the schedule and manage recorded audio. We’ll see this realized later in the demonstration of the Windows platform implementation of Hoot.

Complete Windows implementation

* **Overview**
  + Why deal with the hassle of moving the schedule to-and-from the box using a flash drive?
  + Why put a box in the field that the user cannot interact with?
  + Why use power-hungry, low performance hardware when modern netbooks offer substantially greater battery life for < $400?
  + Why not integrate the entire system into the GUI? Setup your schedule, click the “process” button, and close the lid on your netbook. Done.
* **Java daemon**
  + Uses a platform-independent schedule processor daemon written in Java
    - Validated on Linux, Windows, and OS X
    - Abstracts the difficulties of simple interaction with audio hardware on various platforms
    - Is highly robust and simple
  + When the user wants to begin processing a schedule, they simply click to start the schedule processor. The rest is handled automatically.
* **Demo**
  + Start the scheduler
  + Ensure audio hardware is plugged in
  + Open a schedule
  + Start the schedule processor
  + Close the netbook
  + Watch the netbook wake/process/sleep
    - Note the laptop won’t sleep until 2 minutes after finishing the event.
* **Extended features**
  + An audio browser allows the user to see recorded audio files as they are created in the GUI
    - Users can play the track, copy the file, or delete the file
  + Saved settings
    - Pre-event wake timer
* **Benefits**
  + The user can install the system on a friendly platform in under 2 minutes.
    - Run setup.exe
    - Set your power profile to sleep in some desired period of time (1 minute)
    - On Windows 7, enable wake timers in your power profile
    - Done
  + Substantially improved battery life
    - Easily provides > 1 week usable field life using hibernate
    - Generally provides 3+ days of usable life using sleep
    - This all depends on how many events you want to fire and how often the machine is waking up
  + Ease of use
    - Keeping the entire solution within one application and on one piece of hardware simplifies the system greatly.
  + Easier configuration of audio devices
    - In Windows, simply plugging in the audio hardware is all the configuration you need
    - In fact, audio hardware is hot-swappable while the scheduler is running
    - Users can tweak microphone levels in the audio control panel instead of hard-coding configuration into the box
  + Integrated power management
    - The ARTiGO boxes do not interface with their battery. The system has no way of knowing if the battery is low and it’s likely the system will simply lose power during use.
  + The fan on the ARTiGO box is incredibly loud. It produces a large amount of audio contamination that a good condenser microphone will pick up very easily