A-level Coursework

# Analysis

## Background

### Introduction

In an attempt to find a problem I could apply my Computer Science Coursework to, I had a discussion with the head of music at our school to see if there were any problems facing the department that could be solved with the application of computer science.

In the discussion, it was revealed that student engagement was a problem within the music department, especially for the younger years. The teacher also explained that a lot of students built up the impression that music was not an amazingly diverse subject, which is contrary to reality. So, the basic requirement was to design a solution that would allow students to get engaged with the subject and demonstrate how diverse the subject could be.

During the discussion, it was decided that automating an instrument to play itself would match many of the criteria.

This would not only meet the criteria mentioned above, but also provide a tool that could be used to teach students how different instruments make different sounds.

### Current System

Student engagement is a problem that does not have a simple solution. My system would be one of many techniques that could be used to improve the overall standard of music lessons in the school.

### Clients and Users

The clients would be the music department at the school.

The users would be the music teachers but also the music students.

### Business case for change

### Overview of requirements

The system I design must be able to:

* Increase student attention in lesson
  + Must be interesting/unusual
  + Must be simple to explain how the music aspect of it works
* Be interactive
  + Have features that students can use to play with
  + Must be robust enough to let the students use and experiment with
* Be easy to set up
  + Students and teachers alike will be using this device and so it need to be simple to set up and use
    - Keep number of cables to plug in to a minimum
    - Keep number of first time set up steps to a minimum (4?)
    - Keep number of sequential set up steps to a minimum (2?)
* Be versatile
  + Allow students to play with as well as providing demonstration for the class
  + Allow for multiple different ways to interact with it
    - Downloading midi files from the internet
    - Plugging into existing midi keyboards and systems

### Technical Requirements

* Easily import MIDI files that have been downloaded or made by the music students
* Easily select what track should be played by the robot
  + GUI with all tacks displayed and a representation of what notes will be played
    - Either sheet music or a graphical score
* Be easy to set up
  + Single cable from computer to device (Power??)
  + No drivers required.
  + No looking for the right port to use
  + Simple, well documented user interface
* Robot needs to be able to play the song that it receives in real time.
* Robot should be compatible with other, existing, MIDI devices.
  + Follows MIDI protocol
  + Be able to be used with at least the current set of keyboards found in the music classrooms
* Robot needs to be able to play notes quickly to allow it to play as many different songs as possible.

## Analysis of the Problem

### Contstraints and limitations

The limitations I face with this project are that the Arduino only has a relatively small memory available to the program (32KB for the script and 2KB of RAM). This means the program I write for the Arduino will need to be very light weight so it can be run on the Arduino.

The Arduino is also slow (Clock runs rather slow in comparison to other machines). This means it may lag behind the MIDI inputs and drop bits of data and commands. This means I need to develop a program that can keep up with the torrent of data that is supplied by the serial input.

The Arduino serial input buffer is also quite small. This means that I would need to keep the data processing quick, so the buffer does not fill up and then miss bits of data and commands.

### Scope of the problem

#### IPSO Tables

##### Ardunio IPSO

|  |  |  |
| --- | --- | --- |
| ISPO | Information | Explanation |
| Input | Receive serial MIDI commands   * Note On * Note Off * All Notes Off | This is required for the Arduino to read instructions to know what to do |
| Process | Interpret commands   * Is it note on, note off, or another instruction? * Does this apply to this instrument? (is the channel number correct) | This would be required so that the Arduino would do the correct action when it is required to do it |
| Output | Move servos   * Look up which servos need to be closed and which ones need to be opened. * Find the position values that would be required to close the servos that should be closed and write that to each of the servos | Each servo would have it’s own value that it would need to be set to to make sure that the hole is covered, but the servo is not over torqueing itself. |
| Output | Set blower speed   * If the command is a note off command, turn the blower off. * If it’s a note on command, look up the blower value that needs to be used for that note, and then set it to that value | Each note requires a different amount of air to play, and so it would be necessary to each blower value for each note so the airflow can be set correctly. |
|  |  |  |
|  |  |  |
|  |  |  |

##### Desktop Program IPSO

|  |  |  |
| --- | --- | --- |
| ISPO | Information | Explanation |
| Input | Input MIDI files   * MIDI format 0 [1,pg134] * MIDI format 1 [1,pg134] * MIDI format 2 [1,pg134] | The program will need to be able to take any format midi files as an input an be able to interpret it. |
| Process | Store loaded file in memory | The program will need to be able to store the midi in the memory so it can be displayed to the user and also sent down the wire as midi commands |
| Process | Display the loaded file to the user   * Show each channel * Show each event that occurs on that channel * Display the events that occur across channels | This will need to display the loaded file to the user so that the user can select the track/channel that they would like to be played down the wire to the Arduino |
| Output | Send the signals to the Arduino   * Wait until the correct delta time * Send the packet to the Arduino over the correct port | The program would need to know what information to send at each event and at each time and also be able to send it to the correct serial port for the device the user has connected |
|  |  |  |
|  |  |  |
|  |  |  |

### Model the System

#### Decomposition

(Structure diagram)

#### System Flow-Charts

#### Data flow diagrams

#### data dictionaries

#### Entity Relationship Diagram

#### Class diagrams

## Objectives

### SMART objectives & evaluation criteria

## Appendix

### Notes from interviews

# Documented Design

# Technical Solution

# Testing

# Evaluation