Team D7 : PianoMan

18500: Project Proposal Presentation

Lizzy Thrasher

Surbhi Inani

Vanessa Hwang

Introduction

A self-learning tool for Piano players.

Reads sheet music of song, then lights up LED system using a teaching module for that song.

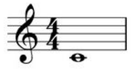
Use Cases

Piano Man
Arranged and Transcribed by William Creamer
Words and Music by Billy Joel
Piano
Pno.

- **Problem Area** : How to make Piano learning more fun and cost efficient ?
- Allows users to scan sheet music of a song and learn to play that song by watching the LEDs
- Will combine Signal Processing (pattern recognition for OMR), Computer Systems and Hardware systems

Requirements and Challenges

- **OMR (Optical Music Recognition)**
    - Requires an ideal scan of a sheet music to convert it to a data structure
        - Little noise, PDF, edges of image are edges of paper, horizontal lines are horizontal, one treble, one bass clef alternating
    - Use OpenCV in Python
    - Convert music data captured to MusicXML

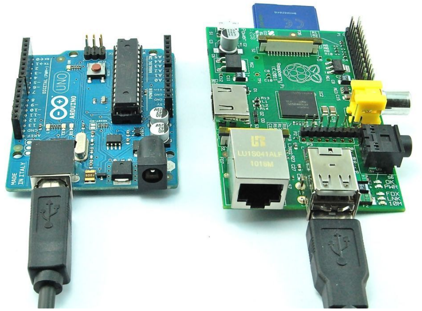


Here it is in MusicXML:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

- **Sheet music data structure in the microcontroller**
    - Output form of OMR -MusicXML sent through wifi
    - Note class - Keys pressed and Time duration/delay

Requirements and Challenges

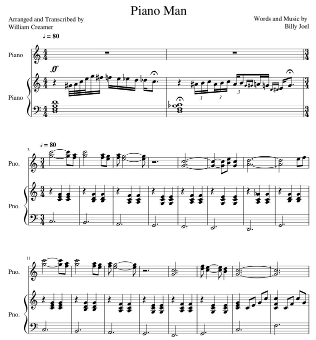


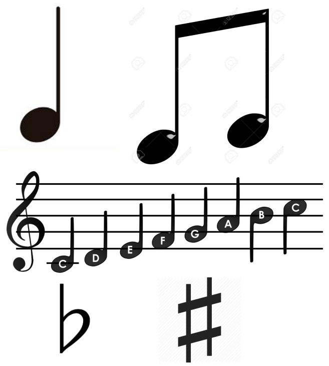
- **Raspberry Pi and Arduinos (Microcontrollers)**
  - Power requirements: +5.1V micro USB supply for Raspberry Pi 3
  - Wifi configuration setup

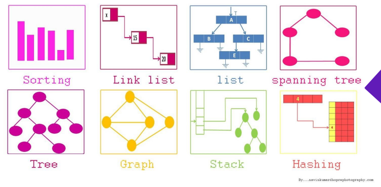
- **LEDs system**
  - Piano white keys are 23.5 mm wide and black keys are 13.7 mm wide. LEDs circuit must be customized to fit these requirements. Each LED coded to one key in the piano.
  - The teaching module should be designed such that each key press is preceded by the LED light indicator in some useful way

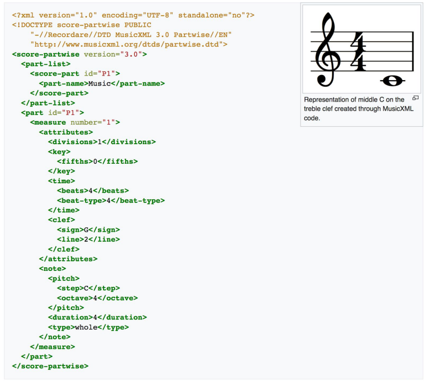
- **Teaching Module**
  - Output from keyboard/MIDI (user input)
  - Should be able to check if the user played the sheet music correctly or not
  - Give feedback to users

Solution Pipeline


Piano Man
Arranged and Transcribed by William Creamer
Words and Music by Billy Joel
Piano
Pno.

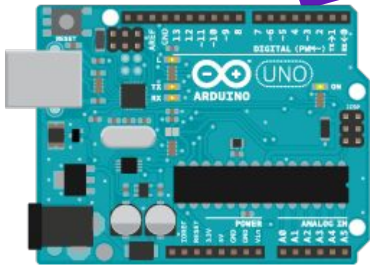






```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

Representation of middle C on the treble clef created through MusicXML code.



Solution Approach - OMR Side

- OMR
  - OpenCV - Python for finding the characters on the sheet music
  - Going to take an ideal scan of sheet music (horizontal lines, little to no noise, binary)
  - Determine pitch and duration of each note
  - Put this data into a MusicXML file (becoming a standard in the music world).

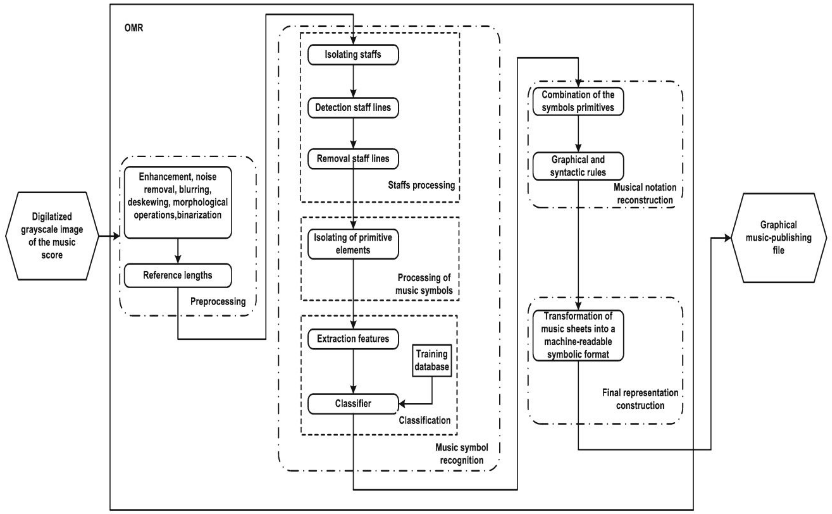

OMR
Digitalized grayscale image of the music score
Enhancement, noise removal, blurring, deskewing, morphological operations,binarization
Reference lengths
Preprocessing
Isolating staffs
Detection staff lines
Removal staff lines
Staffs processing
Isolating of primitive elements
Processing of music symbols
Extraction features
Training database
Classifier
Classification
Music symbol recognition
Combination of the symbols primitives
Graphical and syntactic rules
Musical notation reconstruction
Transformation of music sheets into a machine-readable symbolic format
Final representation construction
Graphical music-publishing file

Fig. 1 Typical architecture of an OMR processing system

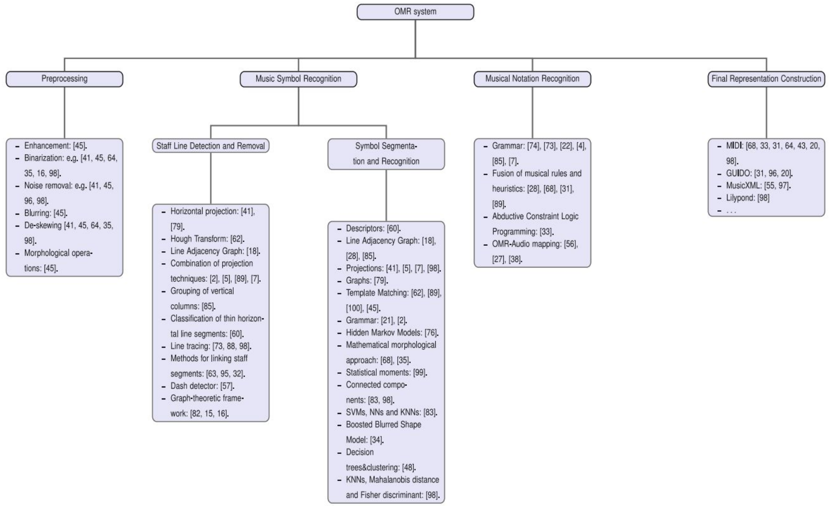
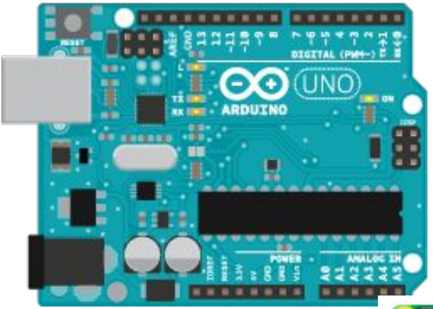
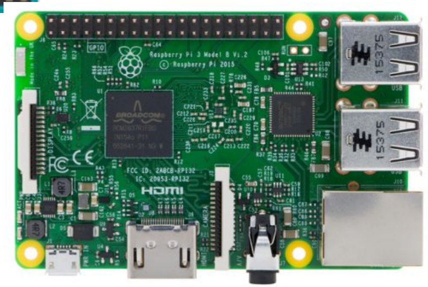
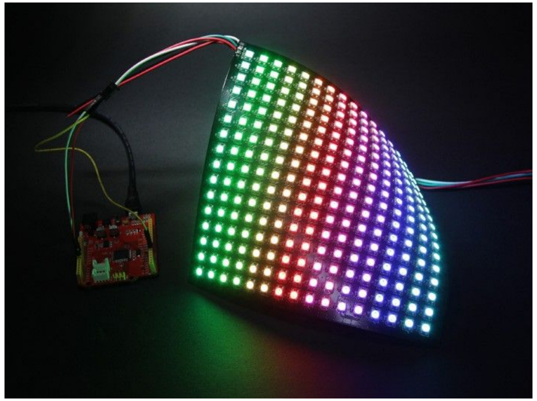

OMR system
Preprocessing
Music Symbol Recognition
Musical Notation Recognition
Final Representation Construction
- Enhancement: [45].
- Binarization: e.g. [41, 45, 64, 35, 16, 98].
- Noise removal: e.g. [41, 45, 96, 98].
- Blurring: [45].
- De-skewing [41, 45, 64, 35, 98].
- Morphological operations: [45].
Staff Line Detection and Removal
- Horizontal projection: [41], [79].
- Hough Transform: [62].
- Line Adjacency Graph: [18].
- Combination of projection techniques: [2], [5], [89], [7].
- Grouping of vertical columns: [85].
- Classification of thin horizontal line segments: [60].
- Line tracing: [73, 88, 98].
- Methods for linking staff segments: [63, 95, 32].
- Dash detector: [57].
- Graph-theoretic framework: [82, 15, 16].
Symbol Segmentation and Recognition
- Descriptors: [60].
- Line Adjacency Graph: [18], [28], [85].
- Projections: [41], [5], [7], [98].
- Graphs: [79].
- Template Matching: [62], [89], [100], [45].
- Grammar: [21], [2].
- Hidden Markov Models: [76].
- Mathematical morphological approach: [68], [35].
- Statistical moments: [99].
- Connected components: [83, 98].
- SVMs, NNs and KNNs: [83].
- Boosted Blurred Shape Model: [34].
- Decision trees&clustering: [48].
- KNNs, Mahalanobis distance and Fisher discriminant: [98].
- Grammar: [74], [73], [22], [4], [85], [7].
- Fusion of musical rules and heuristics: [28], [68], [31], [89].
- Abductive Constraint Logic Programming: [33].
- OMR-Audio mapping: [56], [27], [38].
- MIDI: [68, 33, 31, 64, 43, 20, 98].
- GUIDO: [31, 96, 20].
- MusicXML: [55, 97].
- Lilypond: [98]
- . . .

Fig. 8 Summary of the most used techniques in an OMR system

References: https://repositorio.inesctec.pt/bitstream/123456789/3303/1/PS-07649.pdf

Solution Approach - Hardware Side

- Receive parsed MusicXML file data
- Convert to data for LEDs
    - Position and duration
- Use MIDI to check for user correctness

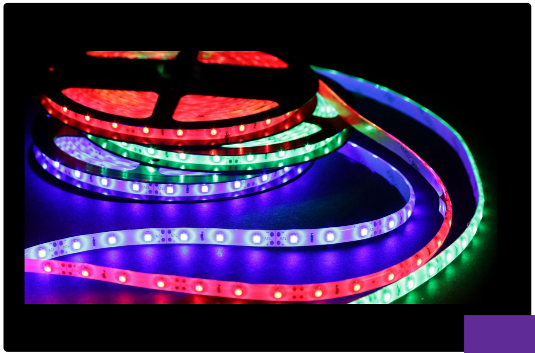
Testing, Verification & Metrics

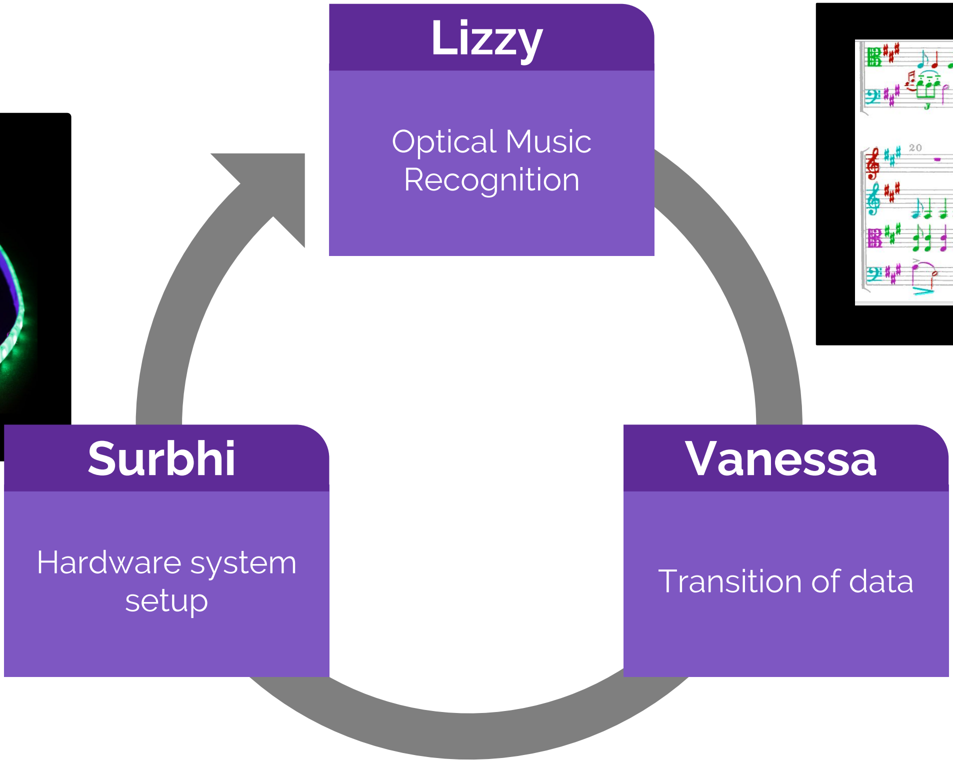
1. **Optical Music Recognition (OMR)**
   Data: Ideal scans of sheet music from MuseScore (https://musescore.com)
   Test:
   1. (a) Use SoundSlice (https://www.soundslice.com) to convert OMR's output MusicXML file to PDF
      (b) Use Notation Software (https://www.notation.com) to play OMR's output MIDI file
   2. Check the difference between original PDF and converted PDF/played MIDI file

2. **Raspberry Pi/Arduino - LEDs**
   Data: MusicXML/MIDI files from MuseScore (https://musescore.com)
   Test:
   1. Test if the microcontroller can successfully transfer data to LEDs
   2. LEDs light up correctly according to the design requirements

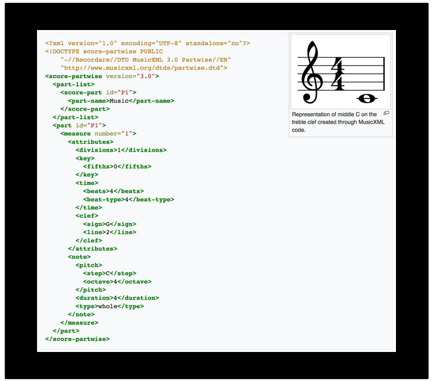
3. **User testing**
   Data: Classmates
   Test:
   1. Let them learn basic songs from MuseScore and collect feedback

Tasks and Division of Labor





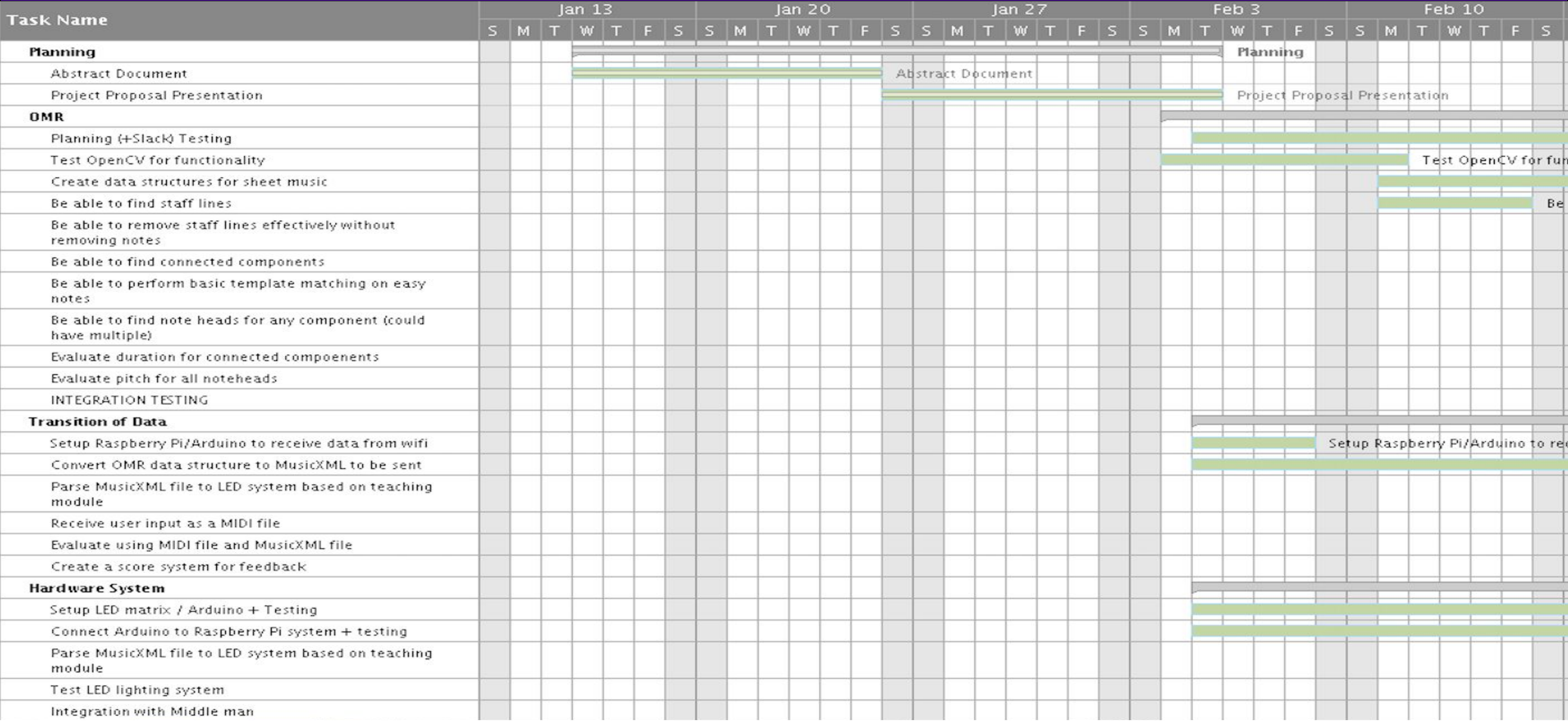
20
p cant. ed espress.
p



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE score-partwise PUBLIC
    "-//Recordare//DTD MusicXML 3.0 Partwise//EN"
    "http://www.musicxml.org/dtds/partwise.dtd">
<score-partwise version="3.0">
  <part-list>
    <score-part id="P1">
      <part-name>Music</part-name>
    </score-part>
  </part-list>
  <part id="P1">
    <measure number="1">
      <attributes>
        <divisions>1</divisions>
        <key>
          <fifths>0</fifths>
        </key>
        <time>
          <beats>4</beats>
          <beat-type>4</beat-type>
        </time>
        <clef>
          <sign>G</sign>
          <line>2</line>
        </clef>
      </attributes>
      <note>
        <pitch>
          <step>C</step>
          <octave>4</octave>
        </pitch>
        <duration>4</duration>
        <type>whole</type>
      </note>
    </measure>
  </part>
</score-partwise>
```

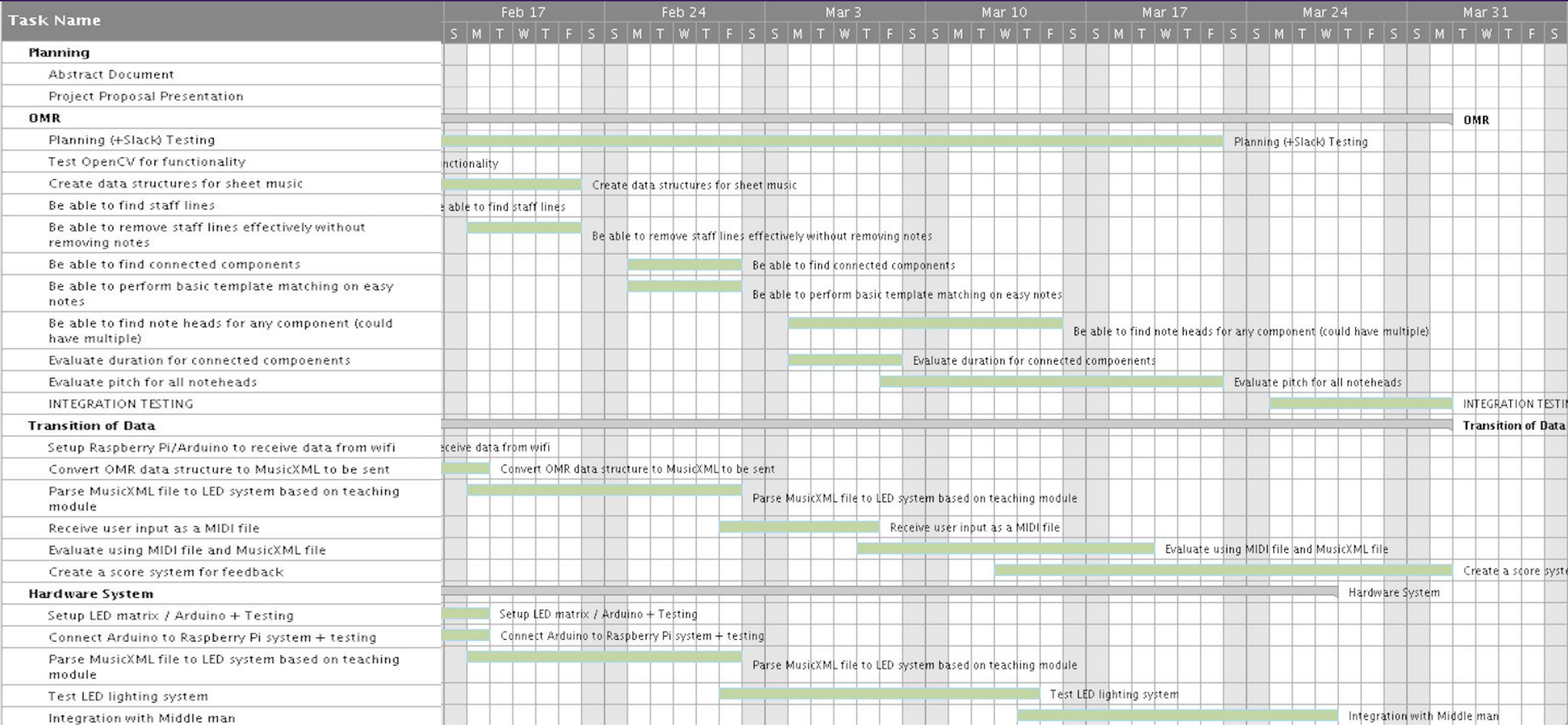
Representation of middle C on the treble clef created through MusicXML code.

Schedule - First Half



Task Name
Planning
Abstract Document
Project Proposal Presentation
OMR
Planning (+Slack) Testing
Test OpenCV for functionality
Create data structures for sheet music
Be able to find staff lines
Be able to remove staff lines effectively without removing notes
Be able to find connected components
Be able to perform basic template matching on easy notes
Be able to find note heads for any component (could have multiple)
Evaluate duration for connected compoenents
Evaluate pitch for all noteheads
INTEGRATION TESTING
Transition of Data
Setup Raspberry Pi/Arduino to receive data from wifi
Convert OMR data structure to MusicXML to be sent
Parse MusicXML file to LED system based on teaching module
Receive user input as a MIDI file
Evaluate using MIDI file and MusicXML file
Create a score system for feedback
Hardware System
Setup LED matrix / Arduino + Testing
Connect Arduino to Raspberry Pi system + testing
Parse MusicXML file to LED system based on teaching module
Test LED lighting system
Integration with Middle man

Schedule - Second Half



Task Name
Planning
Abstract Document
Project Proposal Presentation
OMR
Planning (+Slack) Testing
Test OpenCV for functionality
Create data structures for sheet music
Be able to find staff lines
Be able to remove staff lines effectively without removing notes
Be able to find connected components
Be able to perform basic template matching on easy notes
Be able to find note heads for any component (could have multiple)
Evaluate duration for connected compoenents
Evaluate pitch for all noteheads
INTEGRATION TESTING
Transition of Data
Setup Raspberry Pi/Arduino to receive data from wifi
Convert OMR data structure to MusicXML to be sent
Parse MusicXML file to LED system based on teaching module
Receive user input as a MIDI file
Evaluate using MIDI file and MusicXML file
Create a score system for feedback
Hardware System
Setup LED matrix / Arduino + Testing
Connect Arduino to Raspberry Pi system + testing
Parse MusicXML file to LED system based on teaching module
Test LED lighting system
Integration with Middle man