WoodwindMania

Authors: Angel Peprah, Judenique Auguste, Vivian Beaudoin

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A digital flute input device (flute controller) that mimics functionality of a real flute, accompanied with an application that allows beginners to learn flute in a cheap and effective way. The controller and app will allow the user to learn fingerings, breath control, and posture while also learning scales, notes, and basic music theory with the web application that will communicate wirelessly with the controller and give live feedback.**

***Index Terms*—Arduino, Bluetooth Packet, Breath Detection, Music, Raspberry Pi, Web Application**

I. Introduction

Learning an instrument can be hard for beginners to pursue, due to high costs associated with it. It can easily cost thousands to gain access to a good quality instrument and consistent high-quality lessons. This can be a discouraging aspect of learning how to play a new instrument, especially if the user simply wants to try out the craft or is not sure about committing for a long period of time.

To help solve this problem, we came up with WoodwindMania, a digital way to learn flute that is more cost effective and easy to use for beginners. This will allow beginners to interact with a flute controller that is similar to an actual flute. Additionally, this will allow for a more seamless transition to a real flute in the case that the user decides that they would like to pursue the instrument more seriously after mastering the basic skills.

The flute controller will have the same dimensions as a real flute and will also have buttons located where they normally would be on an actual flute. In addition, the user will have to blow into the device using correct technique to create a noise. Lastly, the device will be tracking its position, so the user will have to be holding it correctly. This will ensure that the user starts to learn the correct skills associated with playing the flute outside of playing the correct notes.

The application will display feedback to the user and allow them to learn correct fingerings of notes from E4 to D6. The application will also teach the user seven scales and test the user on them as well. In the case that the use just wants to play without feedback, there will be a mode for that also.

In terms of competing technologies, there are no direct products that are aimed at beginners. There are electronic wind controllers that output Musical Instrument Digital Interface (MIDI) that are aimed at musicians who want to add wind instruments to their projects, but they do not teach the user anything about the instrument or give feedback to the user about what is being played.

The main goal of this project is to create a budget-friendly system that allows beginners to learn how to play flute in a comprehensive way. Users will be able to learn this new instrument conveniently and at their own pace.

II. Use-Case Requirements

A. Accuracy

The user is looking to use the flute controller to replace a traditional instrument. Therefore, we expect the accuracy of our system to be reasonably high. The accuracy of feedback for fingerings, breath control, and orientation must be greater than 90%.

B. Speed

The input from the user will be sent wirelessly from the onboard Arduino Nano on the flute controller to a Raspberry Pi that will process the user input and host the web app locally. The combined latency of Arduino processing, Bluetooth communication, and RPi processing between the user input and displaying feedback in the application must not be longer than 500 milliseconds. This corresponds to a playing speed of 120 beats per minute, which is the upper-end speed of what we expect a user would play.

C. User Experience

The dimensions of the flute controller must be as close to a real flute as possible. This means it must measure 26 inches in length and a 1-inch diameter. It also must be around 1.3 pounds.

Lastly, user satisfaction (collected in the form of a survey) must be greater than an average score of 4/5, or 80%. Additionally, the device and application should be very beginner-friendly, with no assumed knowledge of the flute or how to play one expected from the users of the device.

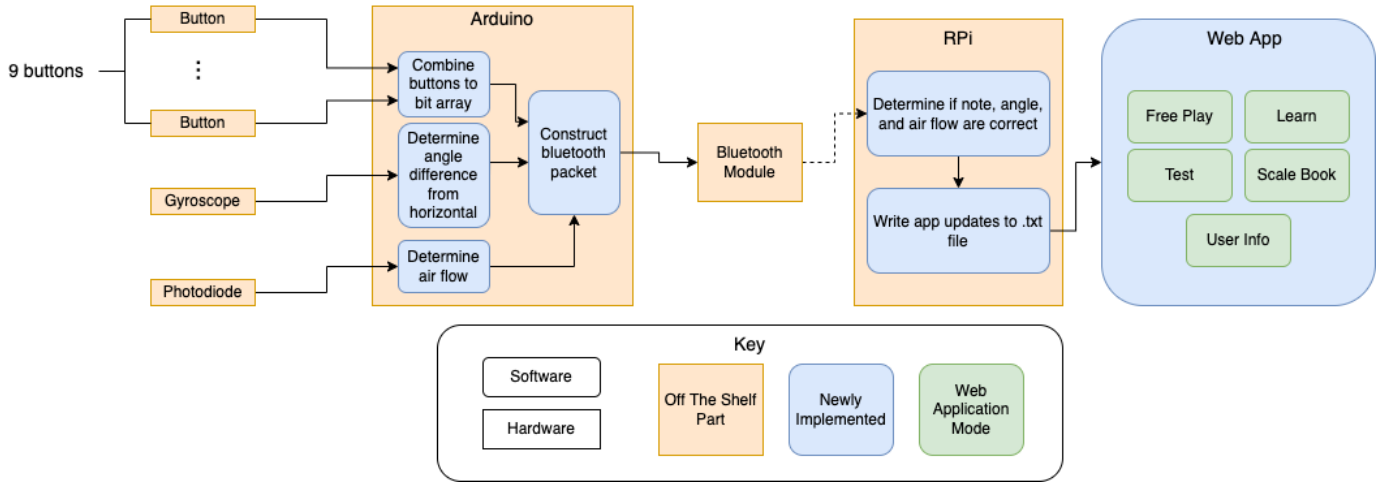


Fig. 1: System Overview

III. Architecture and/or Principle of Operation

There are three main systems in the design of WoodwindMania: the physical controller, the wireless communication, and the web application. The block diagram in Fig. 1 represents the overall system architecture. The controller captures the user input of the fingerings, orientation, and breath control and then sends this data wirelessly to a Raspberry Pi using BLE communication. The RPi then receives this data and plays the corresponding note based on the fingering and breath control amplitude. It will then send the data to the web application where the web app uses the data for feedback.

A. Physical Controller

Our physical controller consists of a collection of sensors encased in a PVC pipe. We will have tactile pushbuttons mounted in the position of the keys on a flute, a gyroscope, and a breath control sensor in the form of a photodiode reading. Our pushbuttons will sense the note fingering the user is playing, our gyroscope will determine the angle at which the user is holding the device, and our breath sensor will determine whether the user is providing enough airflow to produce the note. The Arduino mounted inside the controller will process all of these analog readings and construct a packet to be sent to the RPi via Bluetooth.

B. Communication

Using BLE, the Arduino Nano will send packets to the RPi using characteristics for each type of data. Once the RPi receives the data it will do some data processing, play the corresponding note, and write to a text file for the web application.

C. User Interface

The web application will get the sensor data from this text file and will display feedback depending on the mode the user chooses to interact with. This sensor data is processed in the three modes on the website: free play, learn, and test. The scale book does not require any information from the physical controller.

IV. Design Requirements

A. Accuracy

Our use-case requires that the feedback the user receives from the project is 90% accurate. This can be broken down into the accuracy of our sensors and the accuracy of our wireless communication.

Our data must be sent wirelessly via Bluetooth with 95% accuracy. This is to compensate for any inaccuracies in our sensor readings. We can calculate the accuracy we have in our Bluetooth communication with the following equation:

$$accuracy = \frac{total\ packets\ sent - dropped\ packets}{total\ packets\ sent} \quad (1)$$

We would like this accuracy to hold from a reasonable distance between the controller and the RPi. To be comfortable for the user, we need this accuracy in wireless communication to hold within a distance of five feet between the controller and the RPi.

To ensure that the device senses note fingerings with a 90% accuracy, our pushbuttons should be 99% accurate. We can calculate the accuracy we have in our pushbuttons with the following equation:

$$accuracy = \frac{tests\ where\ push\ button\ presses\ were\ detected}{total\ tests} \quad (2)$$

This should not be a difficult metric to achieve, as pushbuttons are mechanical devices that are rarely inaccurate. Combined with a 95% accuracy of wireless communication, our note fingering detection should hit our 90% accuracy requirement.

To ensure that our project senses correct playing posture with 90% accuracy, our gyroscope should sense a change in 1° with respect to the horizontal. This corresponds to a 95% accuracy, which combined with the accuracy of the wireless communication, should yield an overall accuracy of 90%.

Our breath control sensor should also determine the octave of the note with 95% accuracy. The flute can produce notes from three octaves, but our project only plans to support notes from two out of those three octaves. This is because the very low and very high notes the flute can play are very difficult, and a beginner would be unlikely to play them on a real flute. Therefore, our breath sensor only needs to differentiate between two breath speeds for each note. Combined with the 95% accuracy of our wireless communication, our breath control feedback should be 90% accurate.

B. Speed

The time between input change into our flute controller and receiving feedback from that change should be no longer than 500 milliseconds. This corresponds to a playing speed of 120 beats per minute, which we believe is a reasonable pace for a beginner to play at.

The longest contributor to our latency will be the latency of the wireless communication between our Arduino and RPi. We wish to have a latency of Bluetooth communication of no longer than 400 milliseconds. This leaves the remaining 100 milliseconds for processing data on the Arduino, processing data on the RPi, and sending updates to the web application. To split up these 100 milliseconds, we should have the latency of our Arduino and RPi signal processing be less than ten milliseconds each, and the latency of updates to the web application from the RPi (reading from a text file) should be less than 80 milliseconds. Combining all these latencies yields an overall latency of 500 milliseconds from a user input change to a feedback display.

C. User Experience

Our project should be easy to use and feel similar to a real flute. We break down our user experience requirements into dimensions and battery life.

The length, width, and weight of the flute controller must be close to 26”, 1”, and 1.3 lbs. respectively. More specifically, the length of our controller should be between 25” and 27”, the width of our controller should be between 0.7” and 1.3”, and the weight of our controller should be between 1.2 lbs. and 1.4 lbs. This is in order to best match the feel of a real flute. For the same reason, the buttons on the controller must feel similar to the buttons on a real flute. Our tactile buttons should be within 10% of the diameter of the buttons on a real flute. Additionally, the position of the mouthpiece must be comparable to the actual instrument. The position of the mouthpiece should land within 5% of the position of a real flute. These constraints for the controller are crucial in order to aid the user in transferring their learned skills to an actual flute and increase the efficacy of our device.

The battery life of the flute controller should last at least three hours. We chose this timeframe as the upper bound of a continuous playing session for a beginner. Therefore, the user does not have to replace the battery in the middle of a practice session.

V. Design Trade Studies

Throughout the design phase of our project, we had to make many design tradeoff decisions to best fit our use case. These design decisions can be divided into three main categories: the choice of our breath detection sensor, the nature of our communication between our physical controller and web application, and the nature of our communication between the RPi and our web application.

A. Breath Detection

Beginners learning to play the flute often feel that learning the correct breath control is the most difficult aspect. As a result, it is important for our project that our breath control sensor is accurate for learning breath control on a real flute. Throughout our ideation process, we determined three important requirements for our breath control sensor are accuracy, ability to fit inside the controller, and ease of connection to the Arduino. We examined these three requirements for each breath control sensor we considered.

1) Microphone

The first sensor we considered was a microphone mounted inside the controller. To use this sensor, we would need to convert the microphone reading to a breath speed measurement. This would involve filtering out noise and mapping the volume measurement to the blow speed. This approach would require signal processing, which no one in our group has much experience in. We would also need to distinguish between correct and incorrect blowing forms. We would need to ignore readings where the user is talking or shouting, as well as readings when the user is blowing across or near the microphone without blowing down into the device.

Due to these complications, we predicted that the microphone would be the least accurate sensor for breath control. To determine whether the user was blowing instead of talking or shouting, our hardware would have to analyze the microphone signal in the frequency domain and determine if the sound spectrum matches that typically formed by a user blowing instead of talking. Since blowing would look similar to noise in the frequency domain, we would be looking for an equal band of high amplitude at many frequencies. However, we might not have a way to determine whether the user was blowing downward into the device or simply blowing nearby. If we used the volume reading from the microphone to determine the distance to the controller, we might inaccurately ignore breath control readings where the user is blowing softly into the mouthpiece.

Our second consideration for this sensor is the ability to fit inside our controller. We plan on using a PVC pipe with a 1" internal diameter to contain all our sensors. Therefore, the width and height of the sensor we use should be less than 1". Table 1 gives a breakdown of the dimensions of the sensors we considered using for breath control. Both the width and height of the microphone we considered using are less than an inch, so we would be able to comfortably fit this sensor inside our controller.

Table 1: Breath Sensor Dimensions, Source [3], [4]

Sensor	Dimensions (in)		
	Width	***Height***	***Length***
Microphone	0.62	0.18	0.95
Physical Fan	1.06	1.06	1
LED + Photodiode	0.38	0.23	0.23

Our final consideration is the ease that the sensor connects to the Arduino, which is the microcontroller we will be using to process our sensor data and send it to the RPi. The connection between the microphone and Arduino is simple, as the microphone only has pins for power, ground, and analog out. Therefore, the microphone would only take up one pin on the Arduino in addition to powering the sensor.

2) Physical Fan

The second sensor we considered was a physical fan the user blew on. This fan would be mounted on top of the mouthpiece of the controller and would rotate when the user blew downwards on it. To sense the airspeed, the Arduino would connect to an IR diode and receiver mounted on each side of the fan. Blowing on the fan would break the beam between the IR diode and receiver, and we could convert the rate at which the beam is interrupted to the speed the user is blowing.

This sensor would have better accuracy than the microphone, as no signal processing is necessary to convert the signal to a speed reading. The user would not be able to move the fan when talking or shouting, but they might be able to move the fan when blowing at an incorrect angle (such as across the mouthpiece instead of down). To determine the speed the user is blowing, we would be able to use a more accurate reading from the interruption of the IR beam, instead of using the volume reading from the microphone.

The difficulty with this sensor is the size. If we created a paper windmill out of a small piece of paper (0.75" x 0.75"), the pinwheel would be 1.06" in diameter, which is larger than what would fit on the inside of the PVC pipe. We could shrink the paper fan more, but it would be hard to work with a piece of paper smaller than 0.75" x 0.75".

Similar to the microphone, the IR sensor would only require an analog input pin on the Arduino, as well as power and ground. The circuit required would be a bit more complicated than the microphone, because both the IR LED and the IR sensor need to be connected to resistors to ensure the correct current is being drawn.

3) LED and Photodiode

The final sensor we considered, and the sensor we plan to use in our project, combines the small, portable size of the microphone and the accuracy of a physical fan. This sensor consists of an LED and a photodiode mounted across from each other on different sides of the mouthpiece. A latex barrier will be mounted overtop of the LED. When the user blows downward, the latex barrier blocks the beam of light between the LED and the photodiode, allowing the controller to sense the airspeed of the blow.

This sensor would be of similar accuracy to the physical fan. The user would not be able to trigger the sensor while talking or shouting. The user also would not be able to blow incorrectly into the sensor to trigger it, as the latex barrier will be mounted horizontally and would need airflow downwards to break the beam. We have done a proof of concept with an LED and a photoresistor, and we have already found it to be able to differentiate between blowing softly into the mouthpiece and blowing harder.

The LED and photodiode required for this sensor are both quite small and would be able to fit inside a mouthpiece-sized hole inside the PVC pipe. Table 1 gives the dimensions of this sensor.

Finally, the connection to the Arduino would be simple. The LED requires a connection to digital output and ground, and the photodiode requires a connection to analog input and ground. This would only take an additional two pins off the Arduino.

B. Bluetooth vs. Wired vs. Wi-Fi Communication

There are many factors to consider when looking at how to facilitate the communication between the flute controller and the RPi. There were many options available to us, each with its own pros and cons, which we will discuss here. In order for the flute controller to feel the most natural and allow the most range of motion for the user, we initially only looked at wireless options.

Bluetooth and Wi-Fi communication were both feasible options for this project, as the Arduino platform is compatible with many external modules that enable these types of communication. Additionally, there are Arduino boards with built-in BLE and Wi-Fi as well.

1) BLE

When looking specifically at BLE, we see that it fulfills many of our requirements. One, it is very energy efficient, especially compared to regular Bluetooth. This is achieved via a combination of longer sleep times and shorter transmission bursts (from seconds/minutes to milliseconds). This helps us achieve our goal of a 3-hour battery life on a single 9V battery. Since we are not constantly streaming large amounts of data (such as music), the data transmission rate for BLE –which

maxes out around 1 Mbit/s—satisfies the design requirements, as we are only sending a couple of bytes twenty times or so a second. The BLE connection is also a direct wireless connection to the RPi.

For cons, there is always a chance of dropped packets or lost data in wireless communication, which could be better managed by recommending the user stay within five feet of the RPi for the smoothest experience.

2) Wi-Fi

We also considered using a Wi-Fi connection and ended up not choosing it as our preferred wireless option. For one, this would require the user to connect the flute controller to their local Wi-Fi connection and depend on that network to transmit data to the RPi. We felt that this was not necessary especially since we are currently planning on hosting the application locally on the RPi. Additionally, more power and data would have to be used/sent compared to BLE, which would send four bytes at a time.

3) Wired Connection

We are also considering a wired connection between the flute controller and the RPi as our mitigation for not getting either of the wireless connections working, and this has its own pros and cons to consider. The biggest downside of this option is the potential to affect the use case for the feel of the controller, as the user's motion would potentially be greatly inhibited by the presence of a cable attached to the end of the flute controller. However, this could be slightly improved by using a longer cable that is a couple of feet long.

In terms of the pros, a wired connection would be the most stable and fastest communication option since it would be the most direct connection to the RPi. This would ensure that our accuracy and latency goals were met in our mitigation case.

After looking at these three options, we decided to go with BLE communication between the flute controller and RPi for the energy benefits and direct connection, with the wired being our backup for reliability.

C. Communication between RPi and Web Application

Sensor data from the controller is first communicated to the RPi and then the web application. This is an essential communication line which means the design choices made are important. One such choice was between hosting on AWS versus hosting locally.

AWS would provide a platform for user interaction, as well as more security and scalability. We ultimately decided not to use AWS and host locally instead. User interaction would add a social media feel to our web application. Registered users would be able to communicate with each other and see others' learning progress. For a first implementation website, we did not feel that user interaction was needed. It is something that we can consider for the future. Similar to user interaction, scalability is not necessary for the initial implementation and would be a valid reason to switch when we add more instruments. AWS's best benefit is security for data-driven projects. While our project is data-driven, we will be able to enforce the safety of our users' learning data, private logins, and progress with Django's built-in protections.

Overall, the main reason for choosing hosting locally over AWS is the latency. Our latency requirement is 500 milliseconds, so the user receives audio and visual feedback in a reasonable amount of time. Hosting on AWS would add an additional latency of 500 milliseconds varying by region. This along with the latency from the Arduino to RPi communication would not be efficient for our system.

VI. System Implementation

Our system implementation, briefly described in section III, is broken down into three sections: physical controller, communication, and user interface. In this section, we describe these subsystems in greater detail.

A. Physical Controller

Our physical controller consists of a collection of sensors encased in a PVC pipe.

To sense the note the user is trying to play, the controller will have nine pushbuttons mounted in the device. These nine buttons will be arranged in the same locations as the buttons and keys on a real flute. The Arduino mounted inside the controller will read the button values and combine them into a 9-bit array to eventually send to the RPi.

To determine whether the user is holding the flute controller at the correct angle, the controller will also have a gyroscope sensor mounted inside. This gyroscope will output angle readings for each of the three axes, sending them to the Arduino. The Arduino will combine those sensor readings to determine the angle difference between how the user is holding the controller and the correct position (parallel to the ground).

To determine whether the user is using the correct embouchure to play the given note and is blowing with sufficient speed to produce the note, we will have a breath sensor also mounted inside the controller. Our breath sensor consists of an LED and a photodiode mounted across the mouthpiece from each other, as well as a latex barrier mounted on top of the LED (Fig. 2). When the user blows downward into the mouthpiece, the latex barrier will curl down and obstruct some of the light entering the photodiode. The analog output from the photodiode will be sent to the Arduino, where the Arduino will determine the airspeed that the user is blowing with.

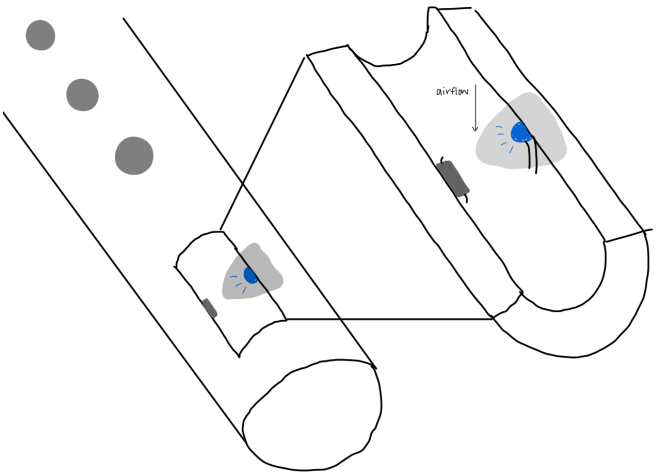


Fig. 2: Breath control sensor

Finally, the Arduino will construct a packet to send to the RPi via Bluetooth. This packet consists of the 9-bit array containing

the buttons currently depressed, the angle with respect to the horizontal that the user is holding the controller, and the airspeed with which the user is blowing.

B. Communication

Once the Arduino Nano creates the bytes representing the flute fingerings, breath control, and position it will broadcast this data as a peripheral device in the BLE scheme, with the RPi acting as the central device. In BLE, members act as either of these types of devices to either generate and post data using services and characteristics or to read/write to said data. That is, the central devices interact with the peripheral devices that generate the data that they want. A diagram of central devices and peripheral devices showing their relationship graphically is located below (Fig. 3).

On the Arduino Nano, we will be making a new service associated with the flute controller, and three characteristics for each type of data that we want to send (fingerings, breath control, orientation). The fingering characteristic will be represented as a short, the breath control will be represented as a byte, and the orientation will also be short.

Once the Arduino updates a characteristic in the flute controller service, a notification is sent out, allowing the RPi to quickly get the updated value when it changes (in a couple of milliseconds). Additionally, the Baud Rate will be set to 9600 bits/s initially, but can easily be scaled up to 115200 bits/s.

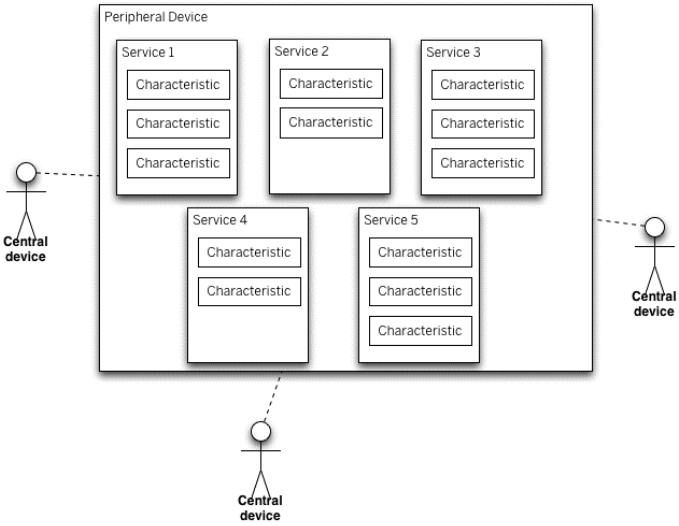


Fig 3: BLE Diagram, Source: [5]

Once the RPi receives a BLE notification from one of the three characteristics, it will then write the updated values to a text file for the purpose of sending the data to the application. It will also determine which note is being played based on the fingerings and breath control amplitude and play it out loud using an audio library, such as SonicPy. This will allow the user to get auditory feedback as quickly as possible. This note mapping will be implemented using a dictionary that is prepopulated with the fingering combinations. If the fingering is invalid, no note will be played.

C. User Interface

The web application is the main view of the user interface. We will use Django, a high-level Python web framework. This framework consists of several languages including HTML, CSS, and JavaScript. The Model-View-Controller system in Django will allow the construction of four different modes for our system: “free play”, “learn”, “test”, and the scale book.

The scale book mode is the simplest of the modes. It will not require the flute controller to be connected. On this page, there will be options to view and listen to the seven major scales that are available to learn. A static flute fingering chart of all the notes is also displayed. And finally, there will be links to resources for additional education.

The purpose of free play is to give the user a space to practice what they learned. It is similar to the experience of playing the flute for fun or practicing a musical piece. The UI for this page will be more relaxed, and there will not be any user feedback. However, it will ensure that the controller is active before starting.

The learn mode is the core of our web application as referenced in Fig. 4. The user will not be able to start if the controller is not enabled. This mode is similar to a private flute lesson. On this page, the user will first choose one of the seven major scales to play. Each scale will go through the notes with displayed flute fingerings. The UI will consist of visuals of the flute fingering with the note on the staff. The user is learning basic music theory through this mode as well. When the data from the user is received, this mode will provide user feedback. These will be displayed as red keys for incorrect hand positions and green keys for the correct hand positions. Additionally, feedback about the overall flute position will show. The user will only receive audio feedback, in the form of a given note, when they use the correct embouchure. This page will have the most computations and will be done by reading a text file with sensor information from the physical controller. Specifically, the Controller, in Django’s MVC system, will have the functions for processing the sensors: the gyroscope, buttons, and breath.

The test mode is where the user will have a chance to show mastery over the seven major scales. The physical controller is required in this mode. The user will first choose the scale to get tested on, and then choose whether to display the scale on a staff. Regardless of choice, there will be a metronome that sets the pace of the scale. The user will play as normal from the “learn” mode, however, if they do not have the correct breath control, there will not be audio. Once the user completes the scale, their score is shown, and with a score of 95, they will pass. The score is calculated based on the correct fingering positions, breath control, and overall flute position. The priority in the test mode is breath control, correct position, and finally, overall flute position. On the processing side, the user inputs from the constantly updating text file are compared with the correct fingering in the scale. As with the learn mode, the sensor information from the breath controller will produce the note. The test results, regardless of a passing score, will be saved inside the user’s database.

To have a complete web application, we will have to

deploy it on a server. We decided to use Apache to process our requests. Additionally, a database is needed to store user information, like usernames, passwords, and progress. There are many databases Django supports, and we have chosen MySQL due to familiarity.

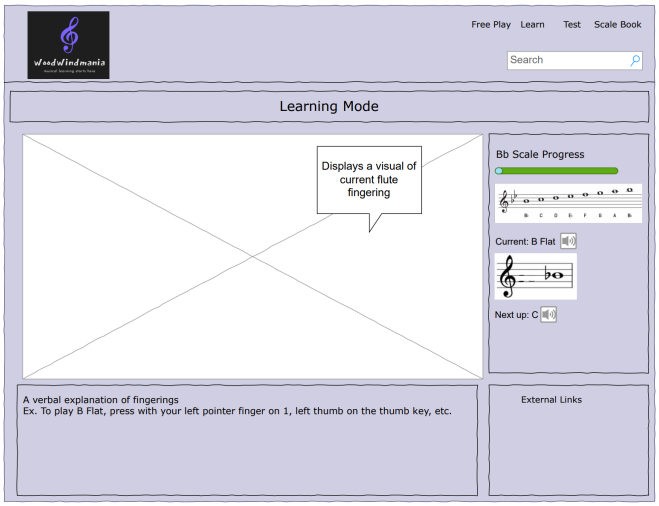


Fig. 4: Mockup of 'Learn' Mode

VII. Test, Verification, and Validation

We separated our use-case requirements into three categories: accuracy, speed, and user experience. Below we break down our testing plan for each requirement.

A. Accuracy

We aim to have 90% accuracy in the feedback from our project. This accuracy relates to the feedback provided for note fingerings, instrument posture, and breath control.

To test the accuracy of our note feedback, we will provide a series of correct and incorrect note fingerings when prompted. For example, when prompted to play the note G5, we will provide inputs that include the correct fingering, a correct fingering for a separate note, and a fingering that does not produce a valid note. We will provide this series of inputs for each note our project supports, aiming for 90% accuracy overall.

Similarly, to test the accuracy of our posture feedback, we will hold the controller at the correct angle (parallel to the ground) and a series of incorrect angles. We will then test that the project visual determines if the user is playing in the correct posture with 90% accuracy.

Finally, we will test the accuracy of our breath control sensor. First, we will test that our sensor does not interpret talking or blowing with incorrect form (such as blowing across the mouthpiece instead of blowing downwards) as an attempt to play a note. Second, we will test that the breath sensor senses the octave the user is playing correctly. We will test this by blowing at the correct speed to produce the note and blowing at the incorrect speed to produce the note (either too soft or too hard). We will compare our blowing speeds to an actual flute so we can determine if our feedback is correct. Again, we aim to reach 90% accuracy on this feedback.

B. Speed

We are aiming for a latency of 500ms between the user producing a note and our project providing feedback. We will test this by recording the time between an input change and visual feedback.

Our input change can come from any of our three sensors. Changes in note fingerings, controller orientation, and breath speed should all provide a visual change in less than 500 milliseconds.

C. User Experience

The primary goal of this project is to replace the cost of buying a real flute. Therefore, we need to continually ensure the feel of our flute controller when adding components. The controller should not weigh too much more, and the size should be similar to the real instrument. We will test this by measuring the length, width, and weight, and ensuring the measurements remain below 26", 1", and 1.3 lbs. respectively.

We also want to ensure that the overall feel of the controller is not too clunky or uncomfortable to learn on. To test this, we will survey beginner flute players and record their rating on the feel of the controller and the likelihood they would use it to practice. We are aiming for 4/5 or 80% of users to feel satisfied using this controller to learn to play the flute.

VIII. Project Management

A. Schedule

Our schedule was based on individual team member responsibilities and how our different components would connect. The physical controller and web application are separate components and can be developed largely in parallel before needing to be tested. The communication chain between the Arduino, RPi, and web application is dependent on partial completion of the physical controller or web application. However, the communication chain can be tested by receiving dummy data or sending dummy data. The first integration milestone is sending correct data from the RPi to the web application. The schedule leaves time near the end of the semester to focus on full integration, testing, and verification.

Refer to Appendix A for our Gantt chart.

B. Team Member Responsibilities

Angel Peprah is primarily responsible for the communication chain in the system. Her responsibilities include sending sensor information from the Arduino in packets to the RPi and transferring that processed information from the RPi to the web application. Angel will work with Vivian to determine what note was played based on fingerings and breath input from the flute controller.

Judenique Auguste is primarily responsible for the construction of the web application and user interface. Her responsibilities include designing the layout of the website, the scales, flute fingerings, and processing user feedback. Judenique will collaborate with Angel to send/process the sensor information in the communication chain between the RPi and the web application.

Vivian Beaudoin is primarily responsible for the construction

of the flute controller. Her responsibilities include crafting the PVC pipe to size, wiring and soldering the buttons, and combining the breath control components. As stated previously, Vivian will work with Angel for note determination.

C. Bill of Materials and Budget

Please see Appendix B for a full list of the equipment we will use and their costs.

D. Risk Mitigation Plans

The largest risk we face in this project is in the communication between the RPi and the Arduino inside the controller. If this communication has high latency, users may begin to feel frustrated using our project. If the communication is unstable, the accuracy of our feedback could be greatly impacted. It is also important to the feel of the controller that the controller remains wireless if possible. We only plan to switch to wired communication if we are unsuccessful in all wireless communication attempts.

To combat this risk, we began researching and prototyping on this area of the project early, and we dedicated a group member's responsibility solely to this communication. We plan on attempting this communication first with Bluetooth, then with Wi-Fi, and finally, with USB wired communication as the last resort. We plan on attempting communication with both Bluetooth and BLE, prioritizing low latency at a short-range (about five feet).

Another risk we face in this project is having inaccurate or inconsistent readings from our sensors. Our physical controller combines a lot of different analog sensors, and therefore we are going to need to do a lot of calibration and testing to find sensors that are accurate and reasonable for our use case. The three sensors we will have inside our flute controller are pushbuttons, a gyroscope, and a breath control sensor. Because a large use-case requirement for our project is the accuracy of our feedback, it is vital that our sensors are accurate.

Our mitigation plan for this risk is to again begin prototyping early and dedicate a group member to focus solely on this area. We will test a variety of different sensors and select the most accurate and consistent sensor that can be calibrated to our needs. For the pushbuttons and gyroscope, we are confident that we will be able to find sensors that are accurate and consistent, as there exist many different sensors we can try and test. The riskiest sensor we plan to use is our breath control sensor, because we are mainly creating this sensor from scratch and breath control is an important part of playing the flute. To combat this, we will begin testing our LED and photodiode breath sensor early, trying a bunch of different subcomponents together (different LEDs, photodiodes, and latex barriers). If we do not have success with this sensor, we plan to use a photoresistor instead of a photodiode. We have already created a working proof-of-concept with this sensor, so we believe we would be able to calibrate this sensor to fit our use-case. We can also use a microphone volume reading as a secondary backup if we run into issues with the photoresistor.

The last risk we have identified for this project is ensuring the feel of our controller is similar to the feel of a real flute. This is important to our use-case, as our project is meant to be a replacement for purchasing a real instrument and getting lessons. If the controller is too heavy or clunky to hold, users may feel uncomfortable practicing on it.

Our plan to mitigate this risk is to try a bunch of different sensors and consistently check the dimensions and comfort of our controller as we add them. There are a variety of different buttons we can use, so we are able to order new ones if we find the size or feel of our current buttons does not match the buttons on the flute. We also plan to compare our controller to a real flute our group has access to, and this should help us adjust the design as we build our controller. We do not think our controller will end up as the incorrect dimensions or weight, because we are using a 1" PVC pipe to contain our electronics and most of our sensors are light. However, if we find that our device is too heavy and cannot change to lighter sensors, we can move our battery and Arduino outside of the controller and use wired communication to the RPi.

IX. Related Work

While researching sensors and requirements for our project, we discovered that there does not currently exist a woodwind controller that accurately mimics a flute. The current flute controller on the market [2] is modeled to be a generic woodwind instrument, so the breath control required is closer to a clarinet or saxophone than a flute. This controller supports output with headphones, a USB cable, or wirelessly to an optional device. This instrument retails for 800€ (~$888) and does not offer any feedback on note playing, posture, or breath control.

Our design will be more specialized for flute playing, including a more accurate breath control sensor, and requiring the user to hold the controller at the correct angle. We will be outputting sound from the RPi wirelessly, and our controller will be significantly less expensive than this one is.

The closest project we have found to our design is a YouTube project [1]. In his design, he uses conductive paint instead of buttons and uses specific buttons to change the octave the flute is currently playing. Our breath control sensor is modeled off his breath sensor, but his flute controller is connected to his laptop via a USB cable and does not provide any feedback.

Our design will use tactile buttons and a wireless controller to resemble the feel of a real flute. We will model our breath control sensor off of this sensor, but we will use the reading of the breath speed to test the user's octave instead of simply changing the note volume.

X. Summary

Our goal is to create an effective woodwind learning tool for beginners to start learning the flute at a low cost. Therefore, it important that our physical controller mimics the playing of a real flute as much as possible, including the dimensions of our instrument, the breath control system, and the feel of the buttons. It is also important that our controller communicates wirelessly to our feedback application and does so quickly and accurately.

Glossary of Acronyms

AWS — Amazon Web Services
BLE — Bluetooth Low Energy
IR — Infrared
LED — Light Emitting Diode
RPi — Raspberry Pi
UI — User Interface

References

[1] KontinuumLab. "KontrolFreak DIY cardboard MIDI flute. Worlds first flute mouthpiece emulator?" *YouTube*, Jun. 29, 2019 [Video file]. Available: https://www.youtube.com/watch?v=HkaP1IJqq98. [Accessed: Mar. 4, 2022].

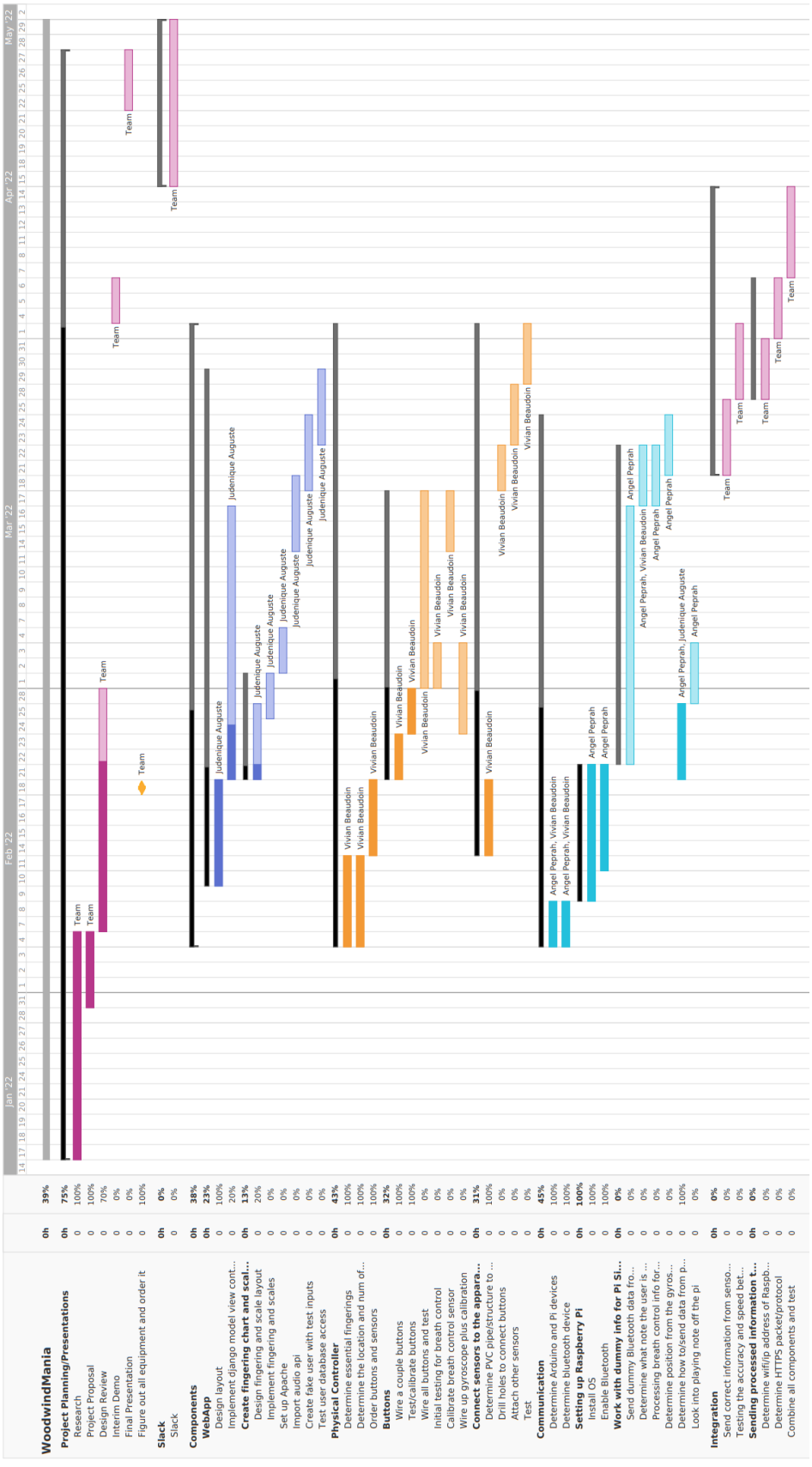
[2] "Buy the Sylphyo – electronic musical instrument – Aodyo." [Online]. Available: https://www.aodyo.com/sylphyo-produit-157.html. [Accessed: Mar. 4, 2022].

[3] "Electret Microphone Amplifier - MAX4466 with Adjustable Gain." [Online]. Available: https://www.adafruit.com/product/1063?gclid=CjwKCAiAo4OQBhBBEiwA5KWu_2fry14Pa8Jg-PhUIAVv299KjdJ6InrgVb7P4s4mAHMuBUoFaqL4AxoCKR0QAvD_BwE. [Accessed: Mar. 4, 2022]

[4] "LED Size Chart: Types & Dimensions," 2022, [Online]. Available: https://evandesigns.com/pages/information-about-led-sizes. [Accessed: Mar. 4, 2022]

[5] SM. 'ble-bulletin-board-model', 2019 [Online Image] Arduino. Available: https://www.arduino.cc/en/Reference/ArduinoBLE?_gl=1*z9dbj6*_ga*MTQyOTAyOTg5Ni4xNjQ0MDIzODk5*_ga_NEXN8H46L5*MTY0NjI3MDU0Ni4zMC4xLjE2NDYyNzA1NDguMA [Accessed: Mar 4, 2022]

Appendix A: Gantt Chart



Appendix B: Bill of Materials

Description	**Model #**	**Manufacturer**	**Quantity**	**Cost @**	**Total**
Raspberry Pi 4	SC15184	Labist	1	$159	Borrowed
16GB Micro SD	SDSQUNC-016G-GN6MA	SanDisk	1	$7.89	$7.89
Arduino Nano 33 BLE	ABX00034	Arduino	1	$22.50	$22.50
3ft PVC Pipe (1 in diameter)		Ventral	1	$12.59	$12.59
16mm Momentary Pushbutton (set of 10)	B07SVTQ7B9	Twidec	1	$7.99	$7.99
Silicon PIN Photodiode (set of 5)	BPW34	Comimark	1	$6.49	$6.49
					$57.46