# Is Mayonnaise an Instrument?

Harry Fernandez, Tomas Vancura, and Min Gun Kim

Department of Electrical and Computer Engineering, Carnegie Mellon University

***Abstract*—A MIDI Controller that generates MIDI messages based on a user's manipulation of household objects in their environment. Our project aims to make sound synthesis and electronic music production more accessible and easier to interface with than standard types of controllers and software interfaces. We accomplish this through an AR headset and a sensing glove which utilize computer vision and signal processing techniques to map the movement of various objects to different MIDI signals.**

***Index Terms*—Accelerometer, Algorithm, Augmented Reality, Camera, Computer Vision, Contact Detection, Deep Neural Network, Design, Display, Force Sensor, Gyroscope, High-Definition Multimedia Interface, Interactive, Inter-Integrated Circuit,, Musical Instrument Digital Interface Controller, Motion Tracking, Music Production, Nvidia Jetson AGX, Object Detection, Object Tracking, OpenCV, Real-Time Processing, Sensor, Single Shot Detector, Sound Synthesis, Universal Serial Bus**

## I. Introduction

Learning sound synthesis and music production can be very intimidating to aspiring musicians who are unfamiliar with unintuitive musical interfaces and the concepts behind them. Studies show that individuals who did not grow up playing a musical instrument struggle much more to learn musical concepts in adulthood and to experiment with music [1]. There are various concepts to learn and understand, and the most common ways of applying these concepts (Synthesizers, Digital Audio Workstations, etc.) can be intimidating. Even expert sound designers struggle when trying to experiment in the studio, as changing parameters requires tediously setting many different knobs and faders within software or hardware interfaces. Oftentimes, unless one really knows what he or she is doing, there is little room for play and creativity in digital music production.

Our vision is to create a new type of MIDI controller that is both intuitive and advanced enough to let anyone experiment with music production. With our project, we hope to broaden the definition of "musical instrument" to include regular, household objects. Our controller, a system that incorporates augmented reality, computer vision, and physical sensors, will allow the user to generate and send MIDI signals to their computer by interacting with their environment in real time. Picking up a cup might generate one sound, while a jar of mayonnaise would produce another. Moving these objects around in space would then change the sound you're generating by augmenting the values of the MIDI signals.

Other XR technologies exist to help with music production, but none of them allow for a tactile experience within the user's own environment. Our approach allows users to use objects that they are already familiar with to explore sound synthesis rather than dropping them into a virtual environment.

Initially we conceived our project as a standalone MIDI Instrument that would be used to both synthesize sounds as well as play them with MIDI notes. However, over the course of development we realized that most people who would benefit from our product probably already own some kind of MIDI device (such as a keyboard), and that our system would in practice function better as a complementary device for controlling effect parameters. Since the glove only requires one hand, the other hand is free to play notes on the keyboard. This means that the glove is solely used to augment the user's sounds rather than as the sole driver of MIDI output. This did not introduce any new requirements, but did loosen some of our initial constraints, such as the need for high note granularity. More information on this decision can be found in Section IV.

## II. Use-Case Requirements

The use case that our system targets is manipulating household objects to create sounds. There are a few requirements that must be met to allow for this.

### *A. Detection & Classification*

The system should be able to detect and classify multiple types of household objects (and therefore controllable parameters) to allow for creative freedom. Having at minimum three detectable objects allows for a reasonable variety of sounds that can be modified in a single session. Additionally, the system only needs to be able to detect objects within less than a one meter radius, as the camera is mounted to the user's head and they are thus limited by the length of their arms. The object detection should be able to successfully and stably classify objects with a success rate of 90% to ensure that the user's interactions produce consistent sound output.

### *B. Latency*

When latency is introduced in electronic musical instruments it can cause resistance for performers making music. Delayed audio feedback is a significant problem in networked music applications, and as such, this system must minimize end-to-end latency. While research shows that some performers can compensate for up to 100 ms of latency, the average performer can only compensate for around 30 ms of latency [2]. Therefore our system should hit this 30 ms target in order to be accessible to a wide range of users. Additionally, contact detection must be very responsive, so the sensing glove must be able to correctly identify contact with an object 99%.

### *C. Motion Tracking*

The sensing glove is responsible for detecting rotational motions of held objects, which will control certain MIDI signals. Ultimately the user will want to feel that their actions have a consistent effect on the sound they produce, so a stable acquisition of positional values is required. It is significant that data is consistent and accurate for the three rotational degrees of freedom: pitch, roll, and yaw. Therefore, our system must be able to track the glove's rotational motion with a standard deviation of 15 degrees based on the population mean of 90 degrees. We believe this metric ensures rotational data to be within the acceptable range of reflecting the user's hand movement.

### *D. MIDI Mapping*

The system's MIDI processor must be able to retrieve signal data from the CV detection output and sensors. It must then quickly translate these signals into MIDI control messages and transmit these messages natively over USB. All of this must be performed fast enough such as to meet the latency requirements described above. That is, the system must behave as any ordinary MIDI controller from the perspective of the user and any of their software.

### *E. Ease-of-Use*

The user must be able to seamlessly use our device with as simple of a learning curve as possible. The device will connect to a display which will stream the camera output along with a GUI. This must be rendered fast enough so that the user does not experience a noticeable delay or become nauseous. The target frame rate is therefore 30 frames per second (fps).

## III. Architecture and/or Principle of Operation

Our system architecture consists of three distinct subsystems. Each subsystem accomplishes a different task in the overall process of translating visual and physical sensor data to MIDI signals. The three components are:

1. Computer Vision (CV)
2. Motion Sensing
3. MIDI Processing

The computer vision component (1) focuses on *object detection* and *potential contact detection.* That is, this process identifies which objects are in the webcam's field of view, and which object, if any, is currently, possibly being touched by the user. The CV component consists of a headset with a mounted webcam (for video input) as shown in Fig. 1. The headset is powered by and communicates with the NVIDIA Jetson Xavier AGX development board (the Jetson). On the Jetson, we perform real-time image processing using a deep neural network to answer the above questions regarding object detection. We use fiducial markers attached to the glove for potential contact detection. This inference information is then sent upon serial request to the MIDI processing component to help determine which type of MIDI signal should be output. We also perform video drawover and display the output to an external display to provide real-time feedback about what the system is doing. Initially we had hoped that this display would be mounted to the helmet itself, but quickly identified that the benefits of this would not be worth the time we would need to invest into getting it working by the project deadline.

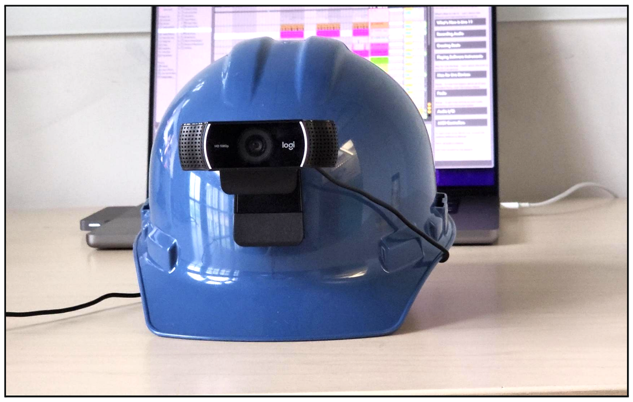

Fig. 1. The final assembled helmet

The motion sensing component (2) focuses on *concrete contact detection* and *motion sensing*. That is, this process identifies whether the user is touching an object as opposed to open air, and how the user rotates that object in space along the three rotational dimensions. The motion sensing component consists of a glove with a force sensor attached to the index-fingertip and an accelerometer and gyrometer mounted on the back of the glove. These sensors are connected to an Arduino Micro (the Micro), which processes the raw sensor data into a packet containing a contact-detection flag and the values of the glove's estimated rotational position. This conversion is done with algorithms based on physics and signal processing that we have implemented. The output packet is sent to the MIDI

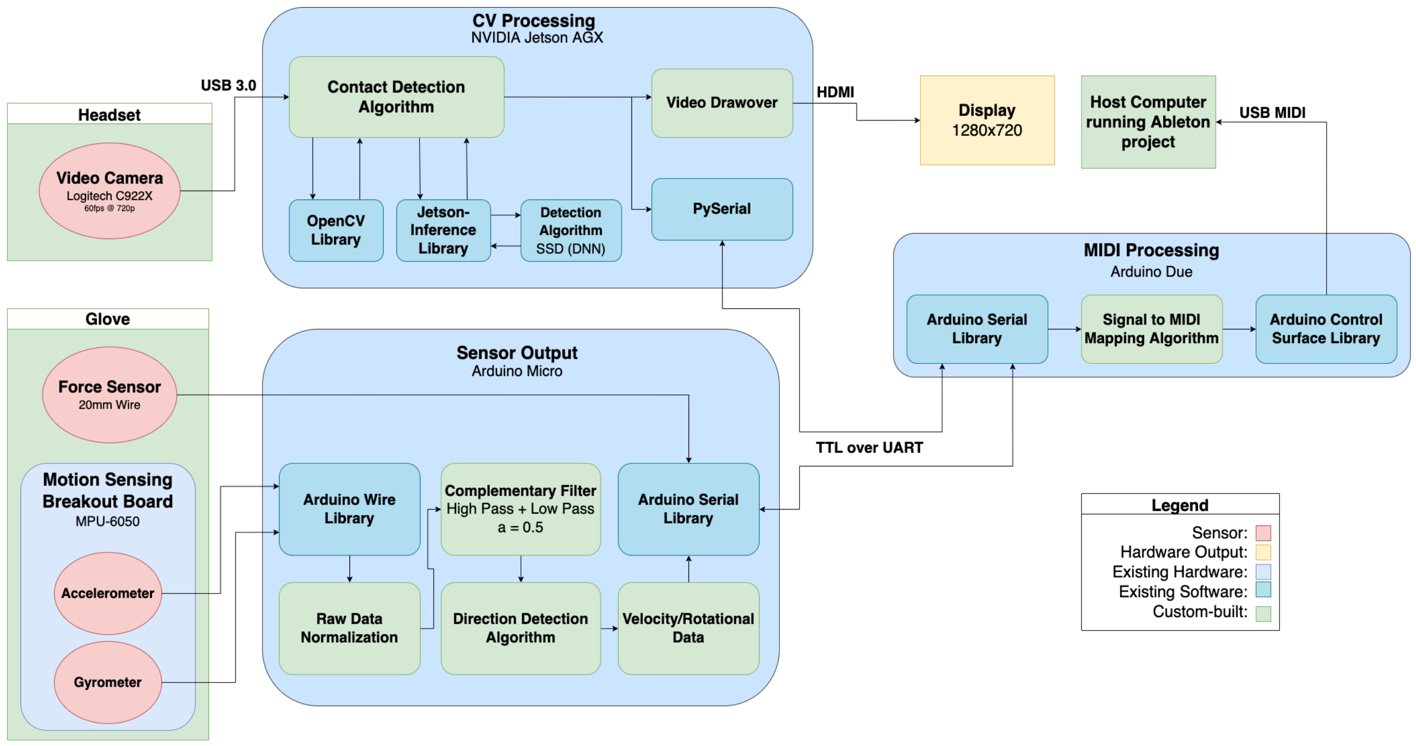

Fig. 2. Full-scale system block diagram, enlarged in Figure 22.

processing component to determine the corresponding MIDI signal parameter values.

Initially, we had planned to track translational data on the Z and Y axes for our MVP and treated rotational data as a bonus feature. However, in testing we discovered that rotational data was more consistent, and we had difficulty getting translational data to produce acceptable results. Additionally, rotational data felt more intuitive to use as it mimics the motion of turning an actual knob. As such we decided to only use rotational data. More information about this decision can be found in Section VII.

The MIDI processing component (3) focuses on *data-to-MIDI translation*. It receives the processed sensor data relating to which object the user might be touching, if the user is touching an object at all, and the estimated position of the glove. The Arduino Due (the Due) maps these values to the appropriate MIDI signal parameters and acts as a USB MIDI output controller.

To demonstrate our project at the ECE Capstone Expo, we drove a pre-configured Ableton Live project with the MIDI outputs from our device. Within this project the system's MIDI signals were mapped in such a way that each object controls the parameters of a totally different sound. In other words, each object can be played as if it is a distinct instrument. Initially we planned to build a custom software synthesizer using Max4Live, but while working on the synthesizer we realized that relying on custom software meant that the final hardware product would not necessarily be as adaptable and easy to use as a general purpose MIDI controller as we had initially hoped.

## IV. Design Requirements

Initially when we had planned to create a standalone MIDI device, we had considered a requirement for note granularity. Note granularity describes the degree to which the user can distinguish between successive beats in time (effectively a measure of temporal resolution of the produced sound). We believed that users would be tapping out individual notes with our product, and as such we needed to detect these impulses occurring within a very short time span (every 0.018 seconds). However, as we shifted away from this use case, we no longer needed this strict requirement. We still decided to enforce our 30 fps metric for video capture, as we still needed clear and sharp frames to perform accurate object detection and hand tracking. The computer vision detection must also be able to operate at these speeds as well.

Another critical requirement of the system is latency. In order to ensure that users are not disoriented by noticeable audio delay the end-to-end latency must hit our target of 30 ms. We expected that computer vision detection and board-to-board communication would be the biggest bottlenecks in hitting this target. In initial testing with the default pre-trained SSD model we observed a worst-case processing time of around 11 ms. In order to give us some slack room to account for the communication between boards we therefore established a 15 ms latency requirement for the acquisition of CV data by the MIDI Processing unit. The performance capabilities of the Jetson combined with clever optimizations of the model used allowed us to reach this target. The communication protocol of the system needs to perform at a high speed as well. Given how little processing should be done for this, a target of 5% maximum latency for communication was established. This works out to a 1.5 ms maximum. The MIDI processing occurs extremely quickly, and as such its impact on the end-to-end latency of the system is considered negligible.

The computer vision detection as stated must be able to distinguish between objects within the frame with a 90% success rate. Accuracy shall be measured through the ability to detect three distinct benchmark objects over various trials. For the sake of variety, we will focus on identifying a cup, a pair of scissors, and a jar of mayonnaise. Additionally, the system must also be able to detect which object is the most likely candidate for being touched with the same degree of accuracy, and its success shall be gauged with the same method.

The force sensor's sensitivity threshold must be precisely tuned such that contact can be detected within our 99% accuracy threshold. The gyroscope broadcasts its output over UART, and as such its data will be broadcast at a rate of up to 10 Mbps. It must be able to extrapolate new pitch, roll, and yaw position accurately enough to meet the 15 degree standard deviation threshold, and this will be measured through successes over various motion trials.

The video streaming and drawover must occur at a framerate of 30 fps at minimum to ensure a smooth viewing experience for the user.

## V. Design Trade Studies

Our project consists of multiple subsystems, each encompassing various design choices and tradeoffs. Here we have detailed our rationale for some of the most important design choices in our system.

### *A. Hardware MIDI vs Software MIDI*

There are two popular ways that a MIDI controller can output signals. A hardware MIDI interface allows signals to be sent over a MIDI cable to devices with MIDI input ports. This approach would allow our MIDI controller to be compatible with hardware synthesizers and instruments. Alternatively, the MIDI protocol can be implemented over USB, and almost all computers can immediately recognize a MIDI device simply by plugging in its USB port. Ultimately we decided to go with a software library that allows the Due to run MIDI over USB. Most musicians who are new to electronic music production will be using a computer to produce music, so software MIDI was chosen to maximize accessibility. As stated in Section II.F, ease-of-use is a huge priority to us, and a plug-and-play product that is compatible with most computers seems like the most sensible approach to meet this goal.

### *B. Communication Protocol*

Our system consists of three computers all communicating with each other over a full-duplex system. As such we needed to decide on a communication protocol. We considered four options: I2C, UART, RS485, and Bluetooth (Table 1). Of these, we decided upon UART.

Inter-Integrated Circuit (I2C) is a full-duplex multi-controller protocol generally used for on-module communication between devices. It has different speed modes, supporting communication at speeds as high as 400 kbps on the Jetson. I2C is also extremely simple to manage, as devices can trade off being controllers or followers simply through the use of STOP and START messages. However, I2C's biggest drawback is its short range of effectiveness. Due to the fact that I2C relies on open-drain current to draw its logic levels, its noise threshold is extremely low. The more cabling in the system the higher the parasitic capacitance gets, and signals can be missed or unreadable altogether.

On the other hand, Universal Asynchronous Receiver-Transmitter (UART) is a hardware interface that allows for bi-directional full-duplex serial communication. It does not require any external hardware, but is generally not rated for long distance communication. It also requires a custom software protocol to be implemented, and arbitration does not work out of the box like with I2C. However, UART is exceptionally fast and can run at up to 2Mbps in some contexts.

The next choice is RS485. This communication protocol is implemented through differential voltage levels, and is generally used in high-noise applications. It runs on the same principles as UART, but requires additional hardware to convert those logic signals into differential signals. It is very stable even at our operating voltage of 3.3-5 V, and is rated for distances of up to 800 m in some applications. It can achieve transfer speeds of up to 10 Mbps, which is extremely promising for our latency requirement. However, even though it is a full duplex protocol, writing software to utilize the protocol can be tricky, as there is still no built-in arbitration like I2C has. It also is not directly supported by most microcontrollers, and instead needs to be driven by hardware TTL-to-RS485 converters running Max 485s.

Finally we have the option of Bluetooth 4.0. Bluetooth has the significant advantage of being wireless, and would allow the user to untether the headset and glove from the rest of the system. Bluetooth can run at speeds of up to 3 Mbps depending on which architecture version is used. It also has an effective range of about 300 ft. However, Bluetooth also does not handle arbitration particularly well, and is more complicated to implement. We would have to consider more frequently dropped packets, as this is quite common with the wireless protocol. Additionally, we still would not be able to make the headset wireless as a USB connection is required to connect the camera to the Jetson.

Ultimately we decided to use baseline UART. While it is possible to increase the transmission distance of I2C using buffers, we were worried that doing so would dramatically increase our latency, and we did not want to force the user to chain multiple buffers off of the glove to the Due. We had initially planned to use RS485, but after attempting to implement it, we had trouble getting the system to perform at high speeds with sufficient stability and synchronization. We had decided not to consider baseline UART due to concerns about noise over the distance we were working with, but after running some tests we realized this was not an issue. With regards to Bluetooth, we felt that supporting a wireless system and the issue of arbitration for multi-directional communication were too complex to make the benefits worthwhile. If it were possible to make the camera wirelessly communicate with the Jetson we would reconsider, but as such Bluetooth does not provide enough of a benefit to outweigh the costs. UART is fast, stable, and works perfectly with our distance range. Ultimately it is the most robust option for our application.

Table 1. Comparison of Communication Protocols

| Name | Protocols | | |
|---|---|---|---|
| | ***Bandwidth*** | ***Maximum Distance*** | ***Arbitration?*** |
| I2C | 400 kbps (Fast Mode) | 8 in. | Yes |
| UART | 2 Mbps | 20-40 ft | No |
| Bluetooth | 3 Mbps | 300 ft | No |
| RS485 | 10 Mbps | 2600 ft | No |

### C. *Serial Arbitration Method*

The custom built software protocol to facilitate board-to-board communication has a huge impact on the latency of the overall system. In choosing to use UART as our communication method we needed to decide how to organize our requests. Since the Due acts as the host in our system, we had to choose between various methods it can use to get data from the other two boards.

The first method we considered was requesting data from both boards on every loop iteration and aggregating the results to generate signal data. This has the benefit of ensuring stability in that no single serial transaction depends on another transaction happening first. However, it has the downside of being slower, as it requires waiting for each end of the transaction before the MIDI signals can be generated.

The second method we considered was requesting data from the Micro every loop iteration, but only requesting data from the Jetson in the event of contact detection. This has the significant benefit of not having to wait for CV detection on every frame, significantly reducing latency. However, this has the potential to propagate any erroneous “blips” in the computer vision data as missed inputs for the system.

Ultimately we decided to proceed with the second method. In the few tests we did run to compare methods the advantages of only running one Jetson transaction per object contact were strongly reflected in our latency results. We managed to account for potentially erroneous data by modifying our potential contact prediction procedure in the computer vision subsystem to stop tracking new objects once contact is detected and only resuming detection once contact is lost.

### D. *Complementary Filter vs Kalman Filter*

MEMS (Micro Electro-Mechanical System) sensors are generally designed to be low-cost, and thus tend to introduce erroneous drift over time. Also, when the spinning axis is aligned with any other axis of freedom it will create gimbal lock. Several filters such as low pass filters, complementary filters, Kalman filters, and Extended Kalman filters can be used to account for this. The Kalman filter is an adaptive filter, which is powerful but has high computational complexity. It uses correlation between prediction and what actually happened to estimate the prediction error. The flow diagram of this filter is divided into four steps as shown in Fig. 3. Firstly the initial value is given, then the prediction step, then the gain of the filter is computed, and finally the estimation is done. The advantage of the Kalman filter is that it can run on devices with very small amounts of memory. On ther other hand, the complementary filter uses a relatively simple algorithm which requires less computation and is easy to implement. Such a feature makes it preferred for embedded systems. High pass and low pass filters are used in coordination to remove accelerometer spikes and gyroscopic drift, respectively.

All of the forces working on the object are measured by the accelerometer, and the long term measurement becomes less reliable as the small forces create disturbances in measurement. So, for the accelerometer, a low pass filter is needed for noise correction. In the gyroscopic sensor, as the integration is done over a period of time, the value starts to drift in the long term. Thus, a high pass filter is needed for gyroscopic data correction. The complementary filter consists of both a low and high pass filter and it is the filter that best suits our system. Fig. 4 is the block diagram of the complementary filter.

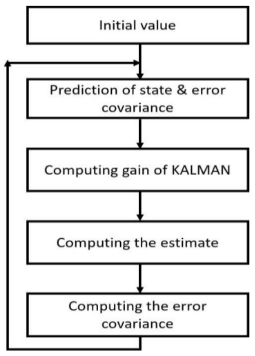
Initial value
Prediction of state & error covariance
Computing gain of KALMAN
Computing the estimate
Computing the error covariance

Fig. 3. Kalman filter flow chart

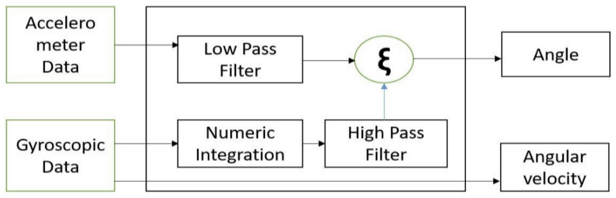

Fig. 4. Block diagram of complementary filter

We have found a research paper [3] that illustrates how Kalman filtering does not provide a good solution to the problem of human posture tracking. Since our project also requires consistent tracking of hand motion, we believe that the usage of complementary filters is ideal for our case. The paper mentions that unlike traditional Kalman filter problems, such as aircraft attitude estimation, the process model and control inputs, are difficult or impossible to estimate. Furthermore, it states that different parts of the human body may require different process models or parameter values. This is a crucial consideration for the case that our project can be further developed in the future to have two gloves. It is important to note that the assumption of a process model governing the motion of a body part causes the Kalman filter to produce incorrect estimates when the process model is inaccurate. Complementary filtering, as it has no assumptions of process dynamics, does not suffer from these problems. Also, complementary filters, due to their low complexity, require significantly less processing resources than Kalman filters. We have found that some applications of Kalman filters require a 32 bit MCU (Microcontroller Unit) and this requirement doesn't fit our decision of using the Micro, which is a 8 bit MCU. Therefore, we will be using complementary filters over Kalman filters.

### *E. Object Detection and Tracking*

Machine learning has become a common approach for object detection in images in recent years due to its robustness and accuracy. Deep learning models using convolutional neural networks, such as the Single Shot Detector (SSD) and YOLO, have proven that object detection can be done in real-time at high frame rates, with sufficiently powerful computer hardware (SSD at 59 fps, YOLO at 45 fps) [4,5]. The drawback of such a method is that the types of objects that can be detected are restricted to the data with which the model was trained.

It is also possible to track objects algorithmically. For example, the Lucas-Kanade Tracking algorithm [6] can detect optical flow in an image to effectively update the position of predetermined tracking points. From prior experience, this approach is computationally efficient in real-time settings, but the drawback is that the tracking points must be either manually or automatically initialized prior to tracking. In our case, this would likely require a calibration phase for tracking each object. Furthermore, the tracking points would be lost should an object leave the frame, and would have to be reconstructed.

Since our use-case involves a moving camera and physical handling of the tracked objects, we determined that robustness and reliability are the most important metrics for detecting and tracking objects. With the Jetson processing board available through the course inventory, we ultimately decided on using the deep learning approach.

### *C. Deep Learning Models for Object Detection*

Time was the main consideration we had when deciding on which deep learning model to use. We concluded that implementing and training our own model, even if based on common architectures, would be infeasible given the timeframe for this project and the other subsystems that would need to be implemented. For that reason, we chose to make use of the *Jetson-Inference* library [7], an AI library for computer vision, provided by NVIDIA and specifically designed for use with Jetson boards. The library provides a pre-trained SSD model with a MobileNet base model. It has been trained on over 300,000 images to detect 91 object classes, including household objects. The library also provides an API for performing inferencing, the step of deep learning where a model is applied to create a prediction.

The drawback of relying on this pretrained model is that we will only have the ability to detect the object types on which it was trained. Transfer learning is an approach for retraining an already existing neural network model on a new dataset with different labels, while taking advantage of many of the already-tuned parameters [7]. Theoretically, this approach can produce a model trained on fewer images, within a shorter amount of time, while retaining much of the accuracy of the parent model. We could thus expand the object classes of our deep learning model on a new dataset, such as a subset of the OpenImages database, which contains 1.9 million images of 600 types of objects [8].

### C. Glove Tracking

The glove can be considered as an object to be tracked by the same neural network used in object detection. However, there are drawbacks to this approach. First, the appearance of the glove is augmented with the sensors and processing boards that are attached to it. Therefore, we would need to retrain the model on a custom dataset including images of the finalized glove design. This would have introduced a new critical path into our schedule. Second, it is possible that the user may pick up an object in such a way that the object they are holding is visible, but the glove is obstructed, and cannot be detected using the network.

ArUco markers are fiducial markers (Fig. 5) that are physically printed NxN binary images used as reference points in AR applications [9]. These markers can be used for overlaying virtual images onto a physical space, or in our case, for tracking an object physically labeled with them. OpenCV provides an API for detecting these markers in an image, which we have tested to run in real-time [10].

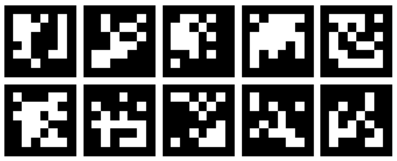

Fig. 5. A set of 10 arUco makers [10]

## VI. System Implementation

Our project is implemented as an AR headset and motion-sensing glove that uses computer vision to convert a user's manipulation of objects in their environment into MIDI signals. A full block diagram of the system is shown in Fig. 2. The fully assembled system is depicted in Fig. 6, and the details of each of the subsystems within the project are detailed below.

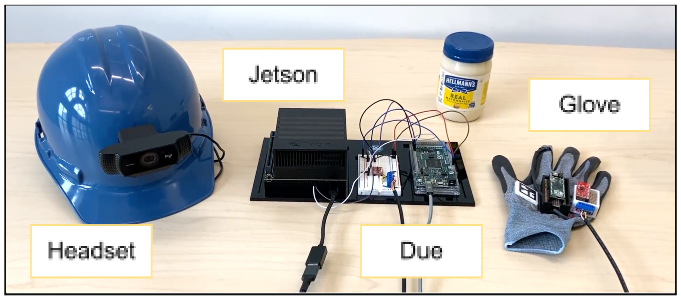

Fig. 6. The fully implemented system

### A. MIDI Processing

All incoming sensor packets are transmitted to the Due. It utilizes the Control Surface library [11] to act as a native MIDI USB device. It uses this library to map the incoming sensor data to MIDI signals. This results in five distinct parameters for our MVP: The ID of the object currently being touched, contact detection, pitch, roll, and yaw (Table 2). Each object has its own distinct set of contact detection and rotational signals, and there are global signals for these parameters as well, for a total of 16 signals. All of these values are output as integers between 0 and 127 with MIDI Control Change messages as per the MIDI protocol.

Table 2. MIDI Signal Mapping Schema

| Sensor Parameter | MIDI Messages | |
|---|---|---|
| | ***MIDI Message*** | ***Message Data Type*** |
| Contact Detected | Note On/Off Event | Binary |
| Object Type ID | Control Change | Integer (0-127) |
| X Position | Control Change | Integer (0-127) |
| Y Position | Control Change | Integer (0-127) |

In order to perform this mapping, the Due constantly requests force sensor data from the Micro. Upon detection of contact, the Micro responds with all current rotational data to the Due until contact ends. Once the Due has received the contact data it requests a numeric ID from the Jetson corresponding to the object it believes is currently being touched. A new MIDI signal for the rotational coordinates will only be generated while contact is actively occurring. Initially we had planned to record the last-known rotational positions of objects so as to preserve their state between contacts, but in testing we discovered that this led to poor accuracy. Instead we reset the rotating frame of reference upon each contact, which has led to more consistent results. All incoming signals from the Micro will be passed through a logic converter as the Micro's operating voltage is 5V and the Due's is 3.3V.

### B. Board-To-Board Communication

The Jetson, Micro, and Due communicate via full duplex UART. All of our processing boards support TTL over UART communication. The Arduino Due acts as the *controller*, and uses its multiple serial ports to facilitate communication with the Micro and the Jetson. The Due requests sensor packets from the Micro and computer vision output packets from the Jetson. As stated in Section V.B UART does not support hardware arbitration, and as such we perform arbitration in software on the Due.

The Micro and the Due do not have the same operating voltage (5V and 3.3V). To resolve this, we route all serial signals on this bus through a 3.3V to 5V *logic stepper* that converts digital signals between these two threshold voltages.

### C. Computer Vision

The CV subsystem performs *object detection* and *potential contact detection*. The goal of this component is to sense and interpret which object the user might currently be interacting with, based on the relative locations of the glove and any objects in the view. As such, *object detection* defines which objects are identified and classified within the webcam's view; *potential contact detection* provides a "best guess" for which object the user might currently be touching, if any.

The hardware for this component consists of a headset mounted with a webcam, as well as the NVIDIA Jetson Xavier AGX, an AI-accelerated CV processing board as shown in Fig. 2.

Real-time object detection is performed using the *SSD-MobileNet* deep neural network, a convolutional neural network designed for embedded computing applications that meets our latency requirements on our hardware. We use an implementation of this network provided by the *Jetson-Inference* library, as well as the included pre-trained classification model [7]. We had planned to improve upon this model by first retraining it on a subset of 6000 images from the Open Images dataset using transfer learning. Retraining the model on these images failed to meet our accuracy requirements, and we decided that improving on it would not be worth the time investment needed. For that reason, we also decided against taking time to collect our own data and merge the datasets.

To account for this and maintain high robustness for object detection, we fixed a lower confidence threshold for the output of the neural network at 30%. This decision however imposes a tradeoff between consistency and accuracy: with a lower threshold, it is more likely that an object will be identified by the neural network, but less likely that it will be assigned the correct class.

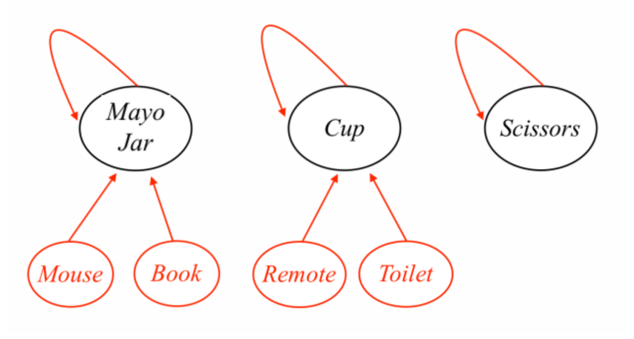

Fig. 7. Object aliasing graph. The mouse and book classes are remapped to the mayo jar class, and the remote and toilet classes are remapped to the cup class.

To mitigate lower accuracy resulting from the lower confidence threshold, we implemented an *object aliasing* procedure for remapping unused objects in our neural network model to our chosen objects that are *consistently and uniquely misclassified* as those objects (Fig. 7). For example, we found that the cup class was often misclassified as a TV remote when in a horizontal orientation, but no other objects were misclassified as a remote; therefore, we could interpret any classification of a remote object as being a cup, since we wouldn't expect the system to see remotes in the first place.

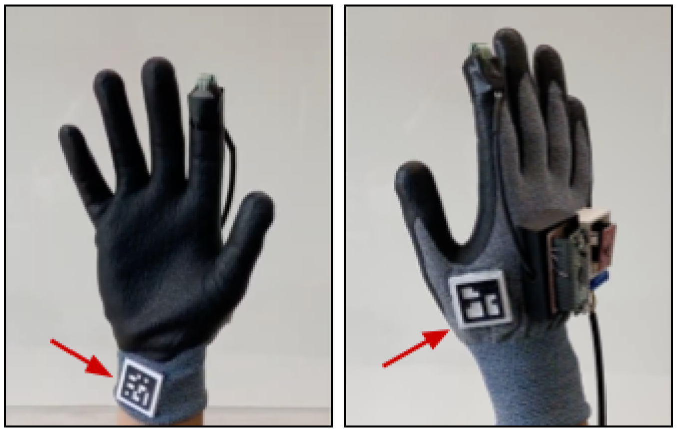

Fig. 8. Two ArUco markers 3D-printed and attached to the glove at multiple locations.

Potential contact detection is implemented by tracking the position of the glove, and identifying the nearest detected object to the glove.

The position of the glove is determined through the frame-by-frame detection of the fiducial markers attached to the glove (Fig. 8). This step is delegated to the OpenCV ArUco library, which performs image segmentation and thresholding to identify square shapes, and validates correct markers by analyzing the inner marker codification [10]. From this information, the pose of the marker (and thus the glove) is estimated, as is the distance between the camera and the glove.

Since the accuracy of object detection depends on varying characteristics such as lighting, orientation, and obstruction, the following *tracking and remembering* procedure is implemented to improve the robustness of potential contact detection.

A dynamically sized region around the glove is determined using the fiducial marker pose information. When this region overlaps with that of a detected object for longer than one second, a potential contact with that object is determined to have occurred (Fig. 9).

After a potential contact is determined, that object is "remembered" (tracked) for the next 5 seconds. After 5 seconds, the tracked object is "forgotten" and tracking resets. While an object is tracked, and the force sensor from the glove is activated, object detection is halted and the currently tracked object continues to be tracked until the force sensor is released. The point of this step is to not falsely detect a new object as being a potential contact while a different object is still being held.

So, when the CV subsystem is polled by the MIDI Processing Unit for an object ID, the currently tracked object's ID (or a null value) is sent as a response.

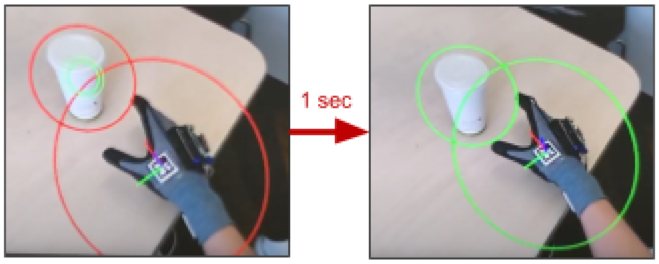

Fig. 9. Visualization of the tracking procedure for detecting potential contact with an object. In the left frame, the cup object is not tracked. After one second of holding the position of the glove, the cup object becomes tracked.

## D. *Motion Sensing*

The goal of this component is to examine whether a user is touching an object or not, and detect any rotational movement of that object. This is crucial for our overall system because contact and movement are the factors that change the sound produced by the software synthesizer.

For the implementation of motion sensing, we used a fingertip force sensor, accelerometer, gyroscope, and the Micro. As shown in Fig. 10 all the sensors are connected to the Micro and the board is placed on the back of the glove. The board is held within the 3D-printed casing that protects the Micro as shown in Fig. 10 This 3D-printed case is attached to the back of the glove using epoxy.

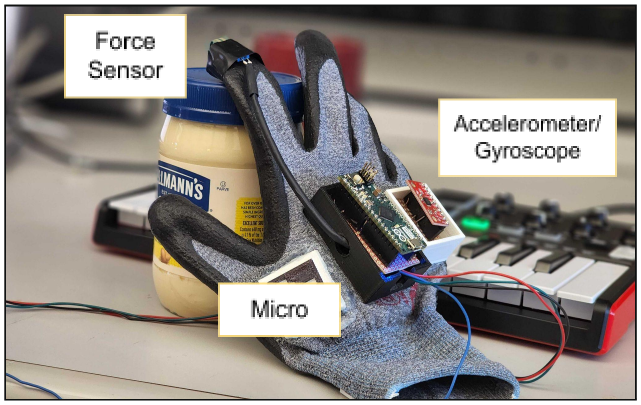

Fig. 10. Arduino Micro and its 3D-printed case mounted on the glove

When implementing a force sensor as shown in Fig. 11, the sensor datasheet [12] suggests that we connect a measuring resistor to maximize the desired force sensitivity range and to limit current. The plot of sensitivity in regards to different resistance values is shown in Fig. 12 According to the graph, we used a resistor with a value of 10K to achieve the widest range of sensitivity.

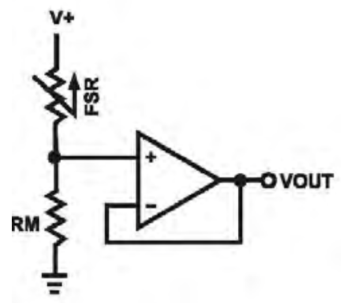

Fig. 11. A simple circuit implementing a force sensor [12]

Initially, the force sensor outputs voltage measures in the range of 0V to 3.3V. The maximum voltage is 3.3V as the supply voltage to the sensor is the 3.3V power supply from the Micro. In order to determine whether an object is touched or not, we considered a threshold voltage of 0.33V. This value is 10% of the supply voltage and we were confident that the value lets us know of the concrete contact with an object. Also, we tested this threshold value during testing and the test mentioned in Section VII confirmed our assumption. Once the output voltage above the threshold is obtained on the Micro, the board converts the data into a binary representation of 0 representing no contact, and 1 representing concrete contact. We did not notice any drift or error with the force sensor data as they are not heavily dependent on the function of time, which is not the case for the accelerometer and gyroscope.

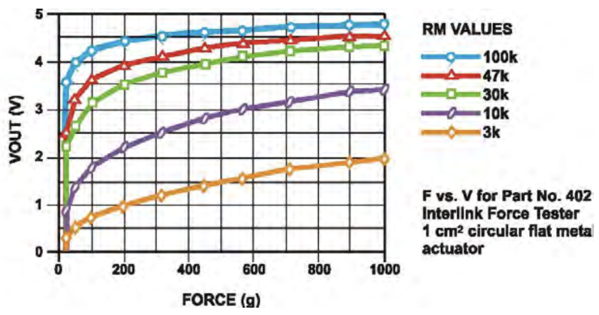

Fig. 12. The plot of sensitivity-based on five measuring resistor values [12]

Raw data is obtained from the gyroscope and accelerometer utilizing the Arduino-MPU-6050 library [13] and is post-processed. Since the accuracy of these parameters are influenced by a number of errors which are a function of time, we first identified the errors in the sensors and applied a signal processing approach to minimize the errors.

The gyroscope drift occurs mainly due to propagated error from the integration of two components: a higher frequency noise variable called angular random walk (ARW) and a slow changing, near-DC variable called bias instability. A good portion of the pitch and roll axis gyroscope drift is removed using accelerometer feedback to monitor position relative to gravity. Filtering the gyroscope output using a low-pass filter canceled a portion of the drift error. The cutoff frequency of the low pass filter was determined by taking the average of output signals and checking its frequency response. We chose the cutoff frequency to be 5 Hz based on the observation that significant information that formulated the rotational data was below 5 Hz. This observation was made based on the frequency response of roll data shown in Fig. 13.

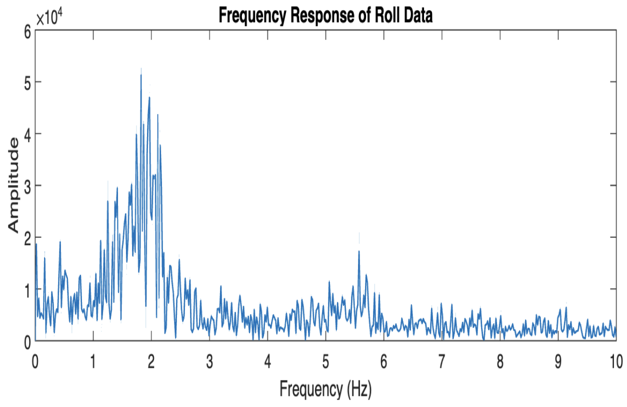

Fig. 13. Frequency response of roll data based on multiple 90 degrees rotation trials.

The filter applied to the rotational data has its characteristics described in Fig. 14 and an example, filtered rotational signal is shown in Fig. 15. This filter is an IIR (Infinite Impulse Response) filter. Our intuition with using IIR over an FIR (Finite Impulse Response) filter was to create a smoother interpolation of rotational position data with a smaller order. Such filters however often do not have a linear phase. In fact, non-linear filters are used widely in noise removal applications, especially in circumstances when there are spike noises - that affects only a small percentage of the samples. In our case, the type of corruption was additive noise resulting from the electronics.

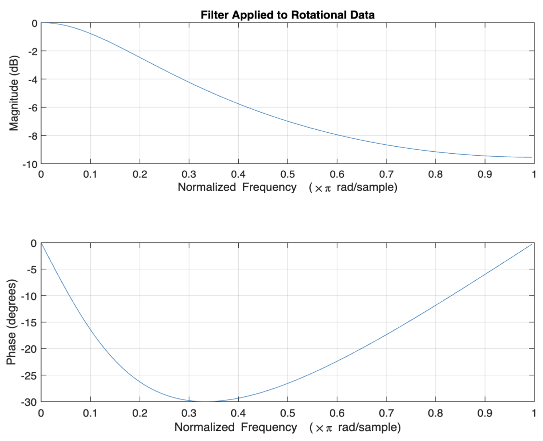

Fig. 14. The magnitude and phase of the filter applied to rotational data.

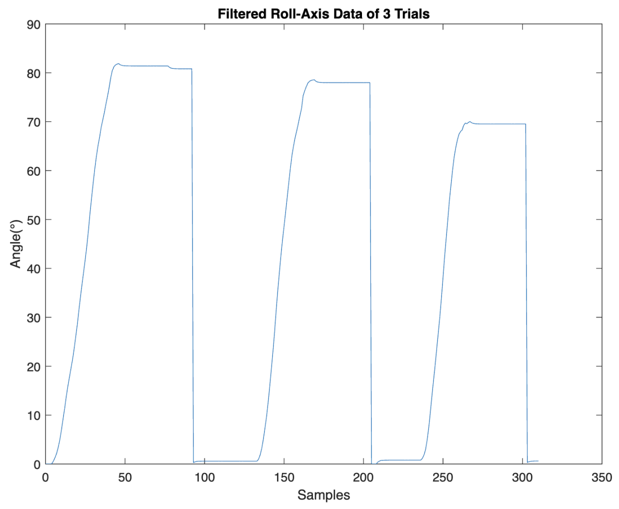

Fig. 15. Filtered roll-axis data of three trials of 90 degrees rotations using the filter shown in Fig. 14.

We cancel long-term drift by implementing a zero angular velocity update to the gyroscope. Any time the device is known to be completely stationary, the gyroscope offset is nulled to zero for that respective axis. We implement this method by resetting the rotational data to zero when there is no contact made with the force sensor. This allows the sensor to correct for the bias instability.

Even though the translational data was not implemented into our glove, we believe that describing our attempts at acquiring and processing translational velocity are worth noting. In the case of an accelerometer, there are three major errors to take into account: constant bias, velocity of random walk, and vibration rectification error (VRE). To tackle the constant bias, the glove is required to have a calibration time of approximately 5 seconds to capture bias occurring when the glove is supposedly in a stable/rest state. This time allows the Micro to measure the average of the initial bias of the output data in the resting state, which can be subtracted in all incoming acceleration data due to hand movement. Then, to account for the velocity of random walk and VRE, we used a moving average filter with a cutoff frequency of 5 Hz and an order of 40 to attenuate signals between the stopband and the passband, which is 0.5 Hz. The low-pass filter was implemented utilizing the BasicLinearAlgebra library [14] removes noise (normally high-frequency) affected by the sensor noise of the electronics. This velocity random walk error is important to be minimized as noise grows proportionally to the square root of time. The filter applied to the acceleration data is shown in Fig. 16. Refer to Fig. 17 and Fig. 18 to observe that the filter denoises the data significantly compared to the original acceleration data.

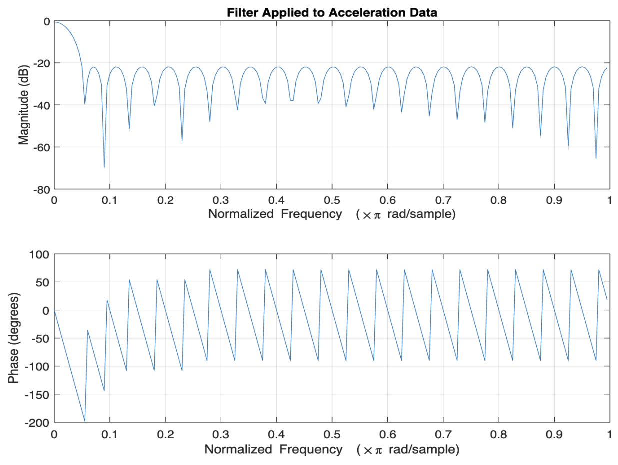

Fig. 16. The magnitude and phase plot of the filter applied to translational acceleration data

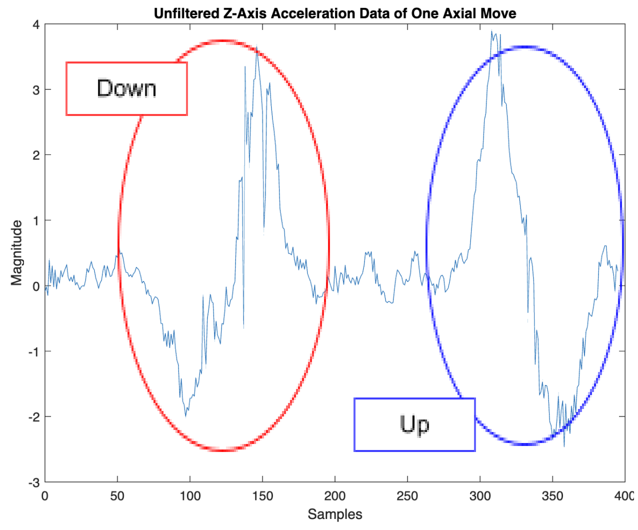

Fig. 17. Unfiltered Z-axis acceleration data for a single up and down movement.

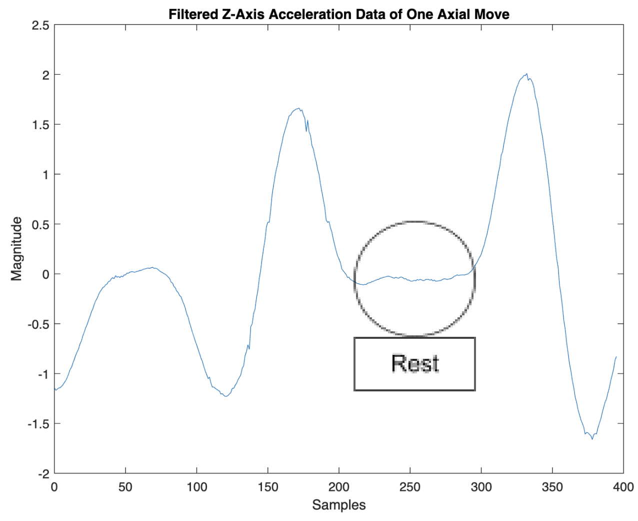

Fig. 18. Filtered Z-axis acceleration data for a single up and down movement.

Once we have the filtered acceleration data, we came up with a method to detect directions per Y and Z axis: left, right, up, and down. Normally, acceleration data looks very similar to a sine curve as a single movement consisting of acceleration and deceleration. An example of acceleration data is shown in Fig. 18 As acceleration data is sinusoidal, we set thresholds to detect directional changes. Often, an up movement will create a positive peak sinusoidal wave converging to zero after reaching the negative peak. A down movement works in a similar fashion, but with the positive and negative peaks switched. To detect directions, we set positive and negative thresholds to check whether a movement was in an up or down direction for the Z axis and left or right direction for the Y axis. This is an important process as we only allow the calculation of integrating acceleration data during the directional movement. If we were to perform integration continuously, the integrated velocity data would drift due to errors in the sensor data.

Using the acceleration data, we integrated them with respect to time once in order to acquire the velocity data. To perform the calculation of integration, we used a method called a Fast Trapezoidal Rule Integrator, which uses the trapezoidal rule to integrate acceleration data and to result them into velocity data. The signal flow graph of this calculation is shown in Fig. 19.

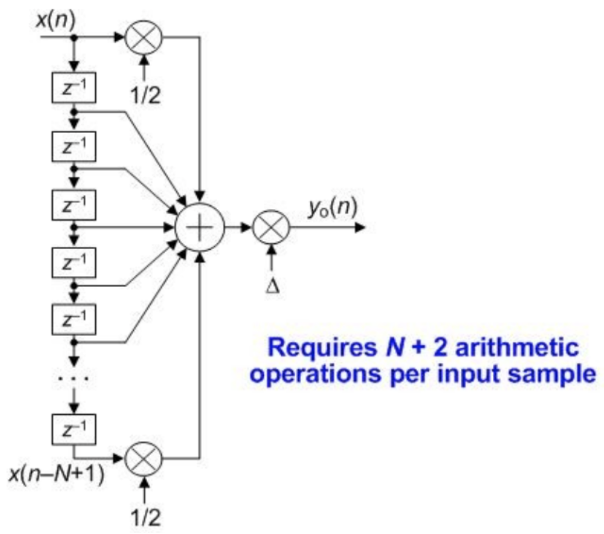

Fig. 19. Signal flow graph of an order N fast trapezoidal rule integrator

We implemented these error reduction methods in the Micro as soon as raw data from all the sensors are captured. As mentioned above, a short calibration period is crucial for more accurate and efficient error correction. After the raw data is processed, acceleration and rotational data are packaged with the binary data from the force sensor to deliver over to the MIDI processing board.

## VII. Test, Verification and Validation

In order to ensure that we meet all of our outline requirements we have defined methods of testing and measurements of success for the various aspects of our project.

### A. *Video Display*

The only metric in our use case and design requirements for the video display relate to the refresh rate of the drawn-over video. We used the *Jetson-Inference* display API to confirm our goal of 30 fps.

### B. *Board-To-Board Communication*

The success of our communication protocol entirely hinges upon its latency and stability. As stated earlier we established a goal of 30 ms for the total end-to-end latency to ensure that system outputs are responsive from the user's perspective. In order to test the latency of our system we recorded timestamps at the beginning and end of each board-to-board transaction. This involved measuring the time from the start of a loop to the acquisition of sensor data from the Micro, then the time between receiving that data and receiving Object ID data from the Jetson, and finally the time taken to package and send this data as MIDI signals to the host computer. We recorded these timestamps over 10,000 trials and averaged the results for each stage of the process. As shown in Table 3, by the end of the project we had more than achieved our goals. The average end-to-end latency ranged from 0.8 ms before MIDI transmission in the best case, and 2.5 ms before MIDI transmission in the worst case. In the worst case scenario, contact is detected and the single request for an Object ID to the Jetson is sent. Because the Jetson may not have the value at the ready, at least one full loop of CV detection can occur at this stage. This leads to a higher potential average time, but this ultimately does not affect end-to-end average latency that much because this case occurs so rarely.

Table 3. Latency Test Results

| ***Subsystem*** | ***Goal*** | ***Actual (Average)*** |
|---|---|---|
| Due ↔ Micro | 1.5 ms | 0.8 ms |
| Due ↔ Jetson | 15 ms | 1 ms |
| Due ↔ Host Computer | 1.5 ms | 0.9 ms |
| End-to-End | 30 ms | 0.8 ms-2.5 ms |

### C. *Object and Potential Contact Detection*

To test the accuracy of object detection, we conducted a series of 9 trials of 60 seconds each. In each trial, we placed one of the three objects in our classification set on a table, and pointed the headset-mounted camera at it from a distance of 0.5 m. The accuracy of object detection is defined as the percentage of frames in which the correct object was identified. To determine the robustness of object detection, we also varied the motion of the camera across the trials. Table 4 summarizes the configuration and results of each trial.

Table 4. Object Detection Trials

| ***Environment Parameter*** | ***Object*** | ***Accuracy*** | ***Average Accuracy*** |
|---|---|---|---|
| Stationary Camera | Coffee Cup | 100% | 99.3% |
| | Mayo Jar | 99% | |
| | Scissors | 99% | |
| Moving Camera | Coffee Cup | 72% | 87.3% |
| | Mayo Jar | 93% | |
| | Scissors | 97% | |
| Object Held in Hand | Coffee Cup | 65% | 67.6% |
| | Mayo Jar | 71% | |
| | Scissors | 67% | |

As can be seen, we met our goal of 90% accuracy for all of our objects with a stationary camera, and we nearly met it for a moving object. We fell far short of this goal when the object was held in the hand, achieving only 67.6% average accuracy. This result was expected, since holding the object can partially obstruct the camera's view of the object, a case that was likely not reflected in the pre-trained neural network model's image dataset. Since picking up and holding an object is nominally our expected use-case, we needed to improve the accuracy for this situation.

The *tracking and remembering* procedure (described in Section VI.D) is our solution to this issue, which effectively loosens the constraint for the held object to be detected in every frame. We repeated the testing procedure described above for picking up an object with this new algorithm to test the accuracy of potential contact prediction. Fig. 20 compares all results across our three objects.

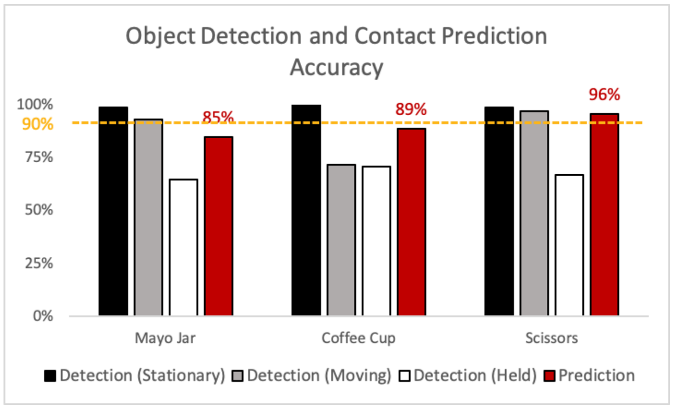

Fig. 20. Contact prediction accuracy improved using the *track and remember* procedure compared to simple object detection.

The average accuracy for potential contact prediction following this procedure is 90%, exactly our target metric. This shows that the *tracking and remembering* procedure was successful in improving the robustness of potential contact prediction as opposed to simply relying on the object detections alone. The coffee cup and mayo jar individually performed slightly below this bar, which can be partly explained by the fact that these two classes were often misidentified as one another from higher angles by the object detection model.

## D. Motion Sensing

We tested two measures to determine the success of motion sensing in our system. First, we checked for force sensor functionality. Since we had electrical tape holding the sensor onto the glove, we needed to consider if the tape was going to obstruct the sensor from capturing data. To test this, we pressed the tester's index fingertip a total of 100 times and checked if the touch was recognized. The test resulted in a 98% success rate of recognizing a fingertip touch, just shy of our 99% goal.

Along with the force sensor, we tested gyroscope sensitivity. When starting the testing procedure, we placed our right hand in front of our body with the palm facing down to calibrate and set a starting point. Then, along the roll-axis, we rotated our hand 90 degrees. We acknowledge that the tester's interpretation of a 90 degrees rotation is subjective, and could vary from trial to trial; however, our goal metric of 15 degrees of standard deviation is large enough to account for this error. The sampled test value was the maximum degree of rotation observed in each trial. After 49 trials, we found out that the standard deviation for roll-axis data with the population mean of 90 degrees was 8.34 degrees. This shows that by fitting our results to a Normal Distribution to our data, one would expect 95% of rotations to fall within $\pm 2 \times 8.34 = 16.68$ degrees. The distribution of all trials is shown in Fig. 21.

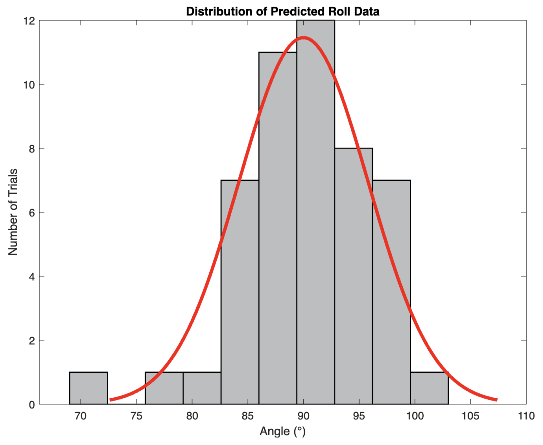

Fig. 21. The distribution of maximum angle per trial of 90 degrees rotation.

## E. User Validation

In addition to quantitative testing to ensure our design requirements were met, we also performed user validation to ensure that all of our use case requirements were met. We reached out to three electronic music major students with varying degrees of proficiency in sound synthesis and had them demo our product. To ensure that each test was the same, we created a preset in the software synthesizer Serum. This preset had two rotational parameters (roll and pitch) mapped to two macro knobs in Serum. These macro knobs can be assigned to any synth parameter within Serum, and are used for extraneous MIDI control. We mapped the Detected Object ID to a third macro knob and set this knob to affect the position of the synthesizer's primary wavetable. This ensured that each object would produce a different waveform for the synthesizer to work with.

After setting up the preset we divided the test into two sections. First, we explained how the system works to the user and pre-mapped our rotational parameter macros to some parameters with very obvious sounding effects. This allowed the users to familiarize themselves with how their actions change the sound. In the second section we removed these pre-mappings and allowed the users to map their rotational macros to whatever they wanted. We interviewed participants both as they were using the product and once again with a Google Form afterwards.

Most of the comments from participants during the validation tests were positive. All of the musicians expressed enjoyment in using our product with the glove's responsiveness being the most liked feature by far. We received comments in real time about the thresholds of both the force sensor and rotation detection, as users were unsure as to how intense their interactions needed to be. A very common observation was an increase in productivity and positive sentiment with the device after roughly 10 minutes acclimation.

We also examined the ergonomics of the glove. We expected that the wire connecting the force sensor and the Arduino may not provide enough room for a person to move their fingers freely. In order to test for any discomfort, we asked our three testers during user validation about how comfortable the glove is to use. All users felt comfortable having the glove on their hand and did not encounter any obstacles when trying to move their fingers. However, we did notice that testers had difficulty taking off the glove, due to the positioning of the circuitry.

Comments from the post-test survey were also helpful. We asked users to report both their enjoyment of the product as well as how straightforward learning to use the product was on a scale of 1 to 5. After analyzing our results we discovered that users overwhelmingly enjoyed the product and that learning how to use it was straightforward (both averaged a 4.5 rating). The most common complaint we received in the survey was that having the external monitor be detached from the helmet was a significant detriment to the experience, as it required the user to be looking in two different directions at once. While we did not have time to implement this given the constraints of the project, after receiving this feedback we've identified it as a high priority for future work. Overall in spite of this complaint our users were very satisfied with our project, with comments stating that they were "so impressed" and that the project was "super cool!" After reviewing all of the feedback, we believe that we have successfully validated our project, even in its current state, to be both satisfying and useful.

## VIII. Project Management

### A. *Schedule*

Our schedule is divided into four sections: Project Logistics, CV Detection Implementation, Sensor Board Implementation, and MIDI Board Implementation. Fig. 22. is the Gantt chart that the team followed and reflects the four sections. Notice some task descriptions have 'T-', which means that the task is related to the testing of a specific component in the system. For all the sections, we acquired and processed data in each individual board before the start of Spring break. After the break, the team focused on communicating that processed data between the boards and integrating the subsystems. Towards the end of the semester, we used our time to add more features after the Interim Demo and perform extensive usability tests.

### B. *Team Member Responsibilities*

Harry worked on configuring the Due as a MIDI device, MIDI output mapping and its signal generation as well as designing the board-to-board communication protocol. Harry also 3D printed the ArUco markers for the glove and made the Ableton project file used for the Final Demo.

Min Gun had the responsibility of implementing the physical sensor processing board. He was also in charge of wiring the Micro and the accelerometer/gyroscope, and assembling the glove. Lastly, he 3D printed the cases for mounting the Micro and the accelerometer/gyroscope.

Tomas worked on CV algorithm implementation on the Jetson. Along with implementation and testing of the algorithm, he assembled the helmet to hold the webcam in position. Furthermore, he used laser-cut in an acrylic tray to hold the Jetson and the Due in place.

### C. *Bill of Materials and Budget*

Table 5 is the bill of materials that keeps track of all the parts that we have purchased and used during research and development, as well as in the final product (highlighted). Our total cost came to $374.45, 62% of the $600 budget.

Table 5. Bill of Materials

| *Item Description* | *Model Number* | *Manufacturer* | *Qty.* | *Cost* |
|---|---|---|---|---|
| Force Sensor (38mm wire) | 30-49649 | Interlink Electronics | 1 | $7.84 |
| Force Sensor (30mm wire) | SEN0297 | DFRobot | 1 | $3.00 |
| Force Sensor (20mm wire) | 34-00004 | Interlink Electronics | 1 | $10.97 |
| 3.3V Voltage Regulator | LD1117V33 | STMicroelectronics | 1 | $3.80 |
| Arduino Due | N/A | Arduino | 1 | $40.30 |
| Arduino Micro | N/A | Arduino | 1 | $20.70 |
| USB A-C Adapter | N/A | Syntech | 1 | $8.99 |
| Accelerometer/Gyroscope breakout board | SEN-11028 | SparkFun | 1 | $29.95 |
| Logitech C922X Webcam | 960-001176 | Logitech | 1 | $79.99 |
| Safety Work Gloves | 100320013 | MUVEEN CO.,LTD | 1 | $12.99 |
| Helmet | HH1000 | Malta Dynamics | 1 | $16.49 |
| Logic Level Shifter | B07LG646VS | KeeYees | 1 | $9.59 |
| I2C Buffer | TCA4307 | Adafruit | 1 | $4.95 |
| 2 x Thin Pressure Sensor | Walfront9snmyvxw25 | Walfront | 1 | $11.09 |
| 5 x Max485 Chip No Arbitration | IC179 | JiuWu | 1 | $6.99 |
| 5 x Max485 Chip W/ Arbitration | TTL to RS485 Module | Songhe | 1 | $7.88 |
| 4 x Pressure Sensors | SEN-09375 | Interlink Electronics | 4 | $43.68 |
| Amphenol Clincher Jacks | COM-14195 | FCI | 5 | $10.50 |

| *Item Description* | *Model Number* | *Manufacturer* | *Qty.* | *Cost* |
|---|---|---|---|---|
| Protoboards | PRT-13268 | SparkFun | 5 | $44.75 |
| Jetson Xavier AGX | N/A | NVIDIA | 1 | Inventory |
| | | | **Subtotal before tax + shipping** | **$374.45** |

### *D. Risk Management*

The first challenge that we faced was not receiving the parts that we needed on time. In particular, there was a shipping delay for the force sensors and IMU sensor. Despite ordering these parts early in advance, we were not able to acquire them until a week prior to Spring break. In order to mitigate this risk, we did our research on working the sensors and prepared all the groundwork before the sensor arrived. Also, we used our own IMU sensor to check if the developed code was working. By the time sensors arrived, we were able to acquire sensor data and work on refining the data.

We faced some difficulties working with accelerometer/gyroscope data. At the beginning, we identified two modes of input: translational acceleration and rotational data. Thus, we wanted to obtain data about an object's translational and rotational movement. Specifically, we wanted to transform acceleration into velocity data for it to be used to manipulate the sonic parameters. During implementation, we noticed instability of detecting directions when calculating velocity. In order to mitigate this risk, we made sure that the rotational data worked for our system. Once we were confident that the rotational data was working, we spent more time trying to refine the velocity data to add it as an extra feature.

For the CV implementation, we planned a step-by-step process figuring out which CV model is the most suitable for our system. In particular, we had the option of either working with a pre-trained or our own model. We scheduled to implement the subsystem with the pre-trained model first, then compare it with that of our own model. By starting with an implementation that we knew would work, we had a fallback in the event that our model did not work well enough. Eventually, we figured out that training our own model was impractical due to lack of time and low accuracy results.

## IX. Ethical Issues

We had multiple discussions about the ethics of our project, whether the existence of our product would have broader implications on society as a whole, and what could result from misuse of our product. Due to the artistic nature of our product as a MIDI instrument we do not believe that our product can intentionally be used to cause harm in a way that is unique to our system. Our system only outputs MIDI data which cannot be used to harm individuals. We discussed the idea of privacy issues given that there is a webcam mounted to the helmet, but felt that this is not a concern since the video data is only made available to the user through the live HDMI output. We also do not believe that there will be any broader societal impacts from the release of our work, as many MIDI controllers already exist and ours does not endanger the existence of other products or industries.

The only real ethical concern that we have involves user safety. In our testing we discovered that it is quite easy to accidentally injure people or things in the user's immediate vicinity. Especially given that one of our currently supported objects is a pair of scissors, we feel that it is extremely important to provide adequate safety warnings to users in the event of an actual product deployment. Users should be reminded to keep their environment clear and that large sweeping motions are not necessary to augment their sounds. While this would constitute misuse of our product, as ethical designers it is important for us to address this possibility.

## X. Related Work

There are two types of already existing technologies that aid music production in immersive environments. First, Sound Playground in VR is a project that allows a user to play pre-determined musical instruments in a virtual setting. The major difference between this project and our controller is whether one is interacting with objects in a virtual reality or in an augmented reality. Our goal is to help users work with tangible objects in the real world, bringing about an intuitive sense to one's sonic experience. Second, Concordia is a musical instrument that allows users to generate and explore transparent sonifications of planetary movements rooted in the musical and mathematical concepts of Johannes Kepler. Concordia highlights harmonic relationships of the solar system through interactive sonic immersion. Unlike Concordia's approach of providing fixed sounds for each planet, our system delivers a wider range of freedom to users as they can map any real objects to MIDI controlled parameters. Our aim is to bring forth a tool that can be used with any software synthesizer to furnish an easy and flexible approach to music production.

## XI. Summary

We created a new class of MIDI controller that utilizes computer vision and various sensors to translate a user's interactions with objects in their environment into MIDI messages. With this project, we set out to enable more musicians to quickly jump into electronic music production, while also providing a tactile and experimental experience for seasoned producers. This second point was validated with students in the Music Technology program at Carnegie Mellon University, who gave positive feedback on both the concept and implementation of our project.

### *A. Future Work*

There are a number of areas of our project in which we could see possible improvement. These areas of improvement largely fall into two categories: *system improvements* and *feature extensions*.

System improvements are the possible solutions to the shortcomings of our project that we have already identified. For example, future work could involve implementing a headset-mounted display, replacing the tabletop monitor used for MVP. A headset-mounted display could make the process of tracking objects feel more natural, contrasted with having to look over a tabletop display. Additionally, given more time, we would attempt to collect our own large dataset of our chosen objects, to improve the robustness of CV tracking, as discussed in section VI.D. Finally, adding tutorialization and safety warnings to our video draw over could address some of the ethical safety concerns discussed in section IX.

Feature extensions are new ideas for improving the functionality of our project that we thought of during development and that were suggested to us during user validation testing. Firstly, tracking translational movements as additional MIDI parameters is a feature that we worked on during the course of the project, but never quite got working to the degree of accuracy and robustness that our use-case requires. Future work would likely involve finishing development in this area. Additionally, we learned that our system could be even more useful in complementing existing digital audio workflows by providing inputs on the glove for toggling a record button or switching between different modes of operation. Finally, implementing and refining Bluetooth connectivity between our system components could be a large improvement for user experience.

### *B. Lessons Learned*

Throughout this project, our team learned the importance and value of proper planning and communication throughout every stage of the design process. By creating extensive back-up plans for each subsystem as part of our risk-mitigation strategy, and by communicating nearly everyday,  we only ever fell behind schedule by one week, and we had plenty of time for refining our system prior to the final demo.

Furthermore, we learned the value of working in an interdisciplinary team such as ours, to which each member brought different experiences. Due to this diversity of thought and background, we were able to come up with solutions to problems that blended our disciplines within ECE.

## Glossary of Acronyms

| | |
|---|---|
| API | Application Programming Interface |
| AR | Augmented Reality |
| ARW | Angular Random Walk |
| FPS | Frames per second |
| CV | Computer vision |
| Due | Arduino Due |
| ECE | Electrical and Computer Engineering |
| FIR | Finite Impulse Response |
| HDMI | High-Definition Multimedia Interface |
| I2C | Inter-Integrated Circuit |
| IIR | Infinite Impulse Response |
| IMU | Inertial Measurement Unit |
| Jetson | NVIDIA Jetson Xavier AGX |
| MCU | Microcontroller Unit |
| MEMS | Micro Electro-Mechanical System |
| Micro | Arduino Micro |
| MIDI | Musical Instrument Digital Interface |
| MVP | Most Viable Product |
| SSD | Single Shot Detector |
| TTL | Transistor-Transistor Logic |
| UART | Universal Asynchronous Receiver-Transmitter |
| USB | Universal serial bus |
| VR | Virtual reality |
| VRE | Vibration Rectification Error |
| XR | Mixed reality |
| YOLO | You Only Look Once |

## References

[1] Wesseldijk, L. W., Mosing, M. A., & Ullén, F. (2020). Why is an early start of training related to musical skills in adulthood? A genetically informative study. *Psychological Science*, *32*(1), 3–13. https://doi.org/10.1177/0956797620959014

[2] Gu, X., Dick, M., Kurtisi, Z., Noyer, U., & Wolf, L. (2005, June). Network-Centric Music Performance: Practice and experiments. *IEEE Communications Magazine*, *43*(6), 86–93. https://doi.org/10.1109/mcom.2005.1452835

[3] Young, A. D. (2009). Comparison of orientation filter algorithms for realtime wireless inertial posture tracking. *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*. https://doi.org/10.1109/bsn.2009.25

[4] Liu W. et al. (2016) SSD: Single Shot MultiBox Detector. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham.

[5] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[6] Lucas, B. D. and Kanade, T. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81, pp. 674–679, 1981

[7] Franklin, D (2022) Jetson-Inference [Source code]. https://github.com/dusty-nv/jetson-inference

[8] *OpenImages V6 - Description*. Open images V6 - description. (n.d.). Retrieved March 4, 2022, from https://storage.googleapis.com/openimages/web/factsfigures.html

[9] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., &amp; Marín-Jiménez, M. J. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition, 47(6), 2280–2292. https://doi.org/10.1016/j.patcog.2014.01.005

[10] OpenCV (2022) [Source code]. https://opencv.org

[11] Control Surface Library (2022) [Source code]. https://github.com/tttapa/Control-Surfa

[12] Interlink Electronics. FSR Integration Guide. Retrieved from https://www.digikey.com/en/pdf/i/interlink-electronics/interlink-electronics-fsr-force-sensing-resistors-integration-guide

[13] Arduino-MPU-6050 (2022) [Source code]. https://github.com/jarzebski/Arduino-MPU6050

[14] BasicLinearAlgebra (2022) [Source code]. https://github.com/tomstewart89/BasicLinearAlgebra

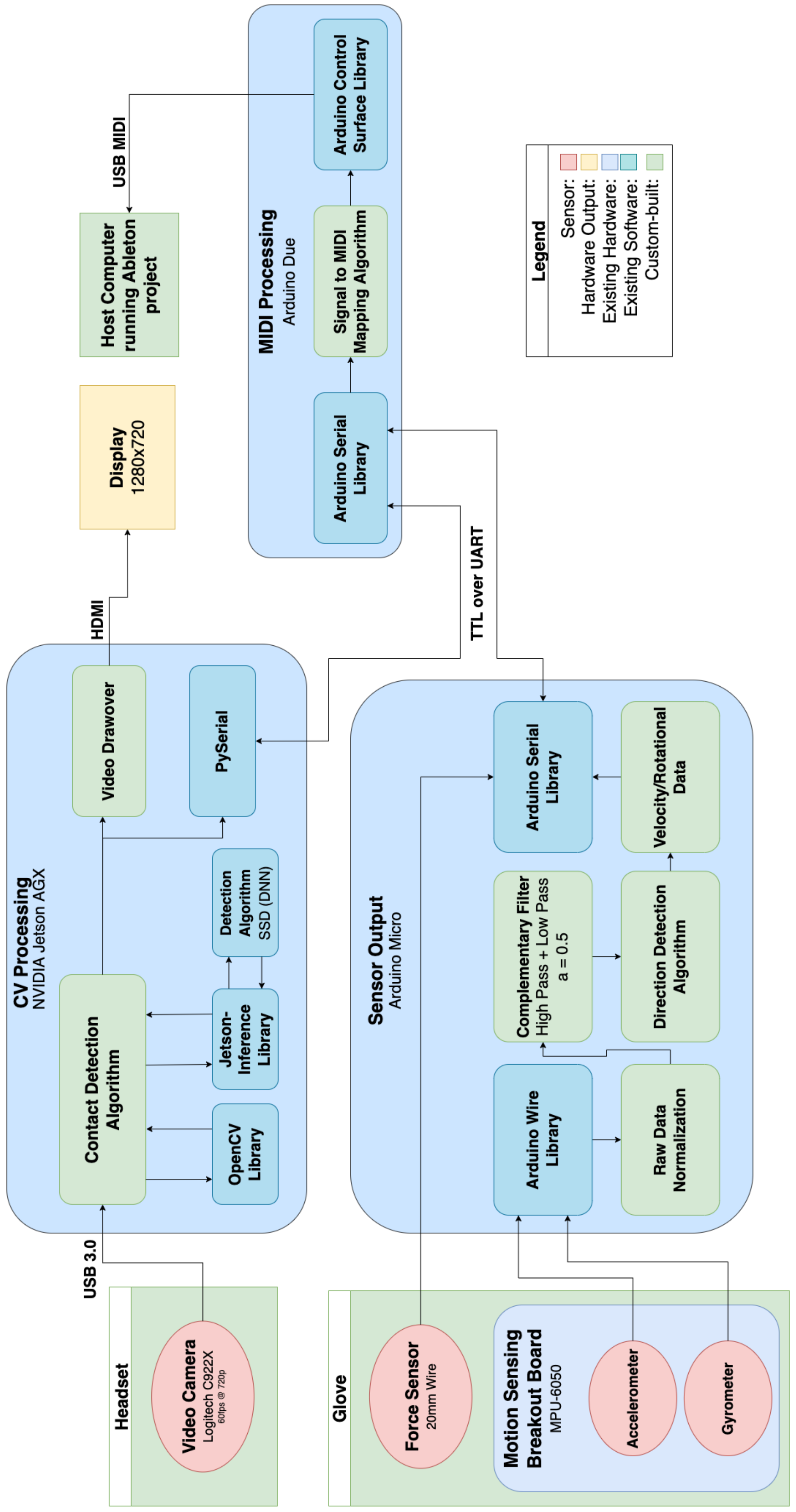

Fig. 22. Full-scale system block diagram

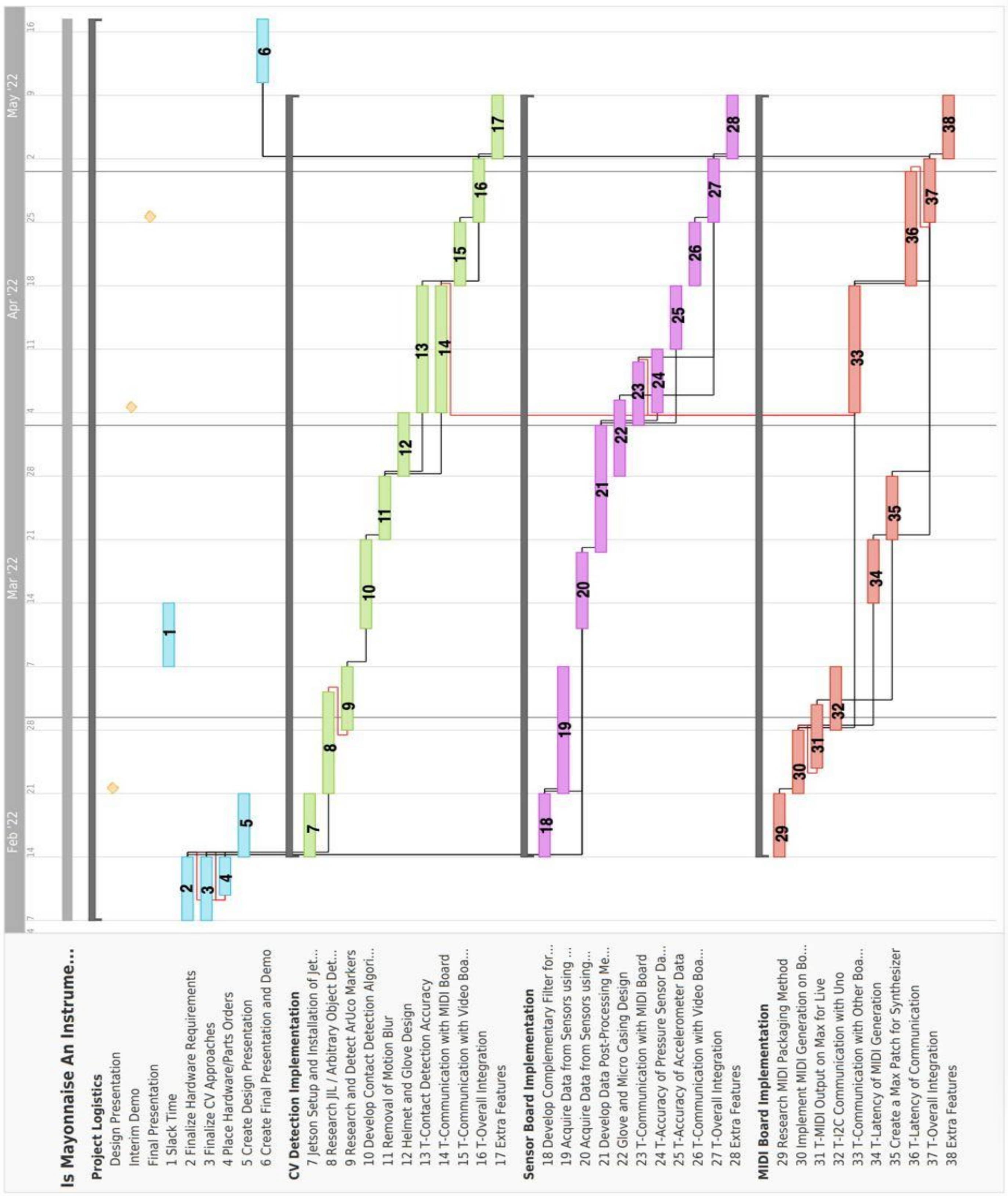

Fig. 23. Final Gantt Chart Schedule with color-coded teammate responsibilities