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. 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, Max4live, Musical Instrument Digital Interface Controller, Motion Tracking, Music Production, Nvidia Jetson AGX, Object Detection, Object Tracking, 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 let you generate and send MIDI signals to your computer by interacting with your environment in real time. Picking up an apple 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 parameters of the MIDI signal.

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.

II. Use-Case Requirements

From these use cases one can extrapolate that the user will want to manipulate various household objects to create sounds, and that they may then wish to create impulses with these sounds in order to play melodies and rhythms. There are a few requirements that must be met to allow for this.

A. Note Granularity

First the note granularity must be considered. Users will play notes by quickly making contact with detected objects. In order to allow for maximum creativity and flexibility, this system must be able to detect independent contact impulses with a reasonable amount of precision. A majority of songs fall well under 200bpm, and most hand-played rhythms will not need notes faster than 1/16th notes. However, 1/8th notes are a reasonable minimum in the event that reaching a granularity of 1/16th is not possible.

B. Detection & Classification

The system should be able to detect and classify as many different household objects within a frame as possible to allow for many degrees of freedom. Having at minimum three objects be detectable at any given time allows for a reasonable variety of sounds that can be generated in a single session. Additionally, the system only needs to be able to detect objects within 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, but ultimately producing any kind of sound is still acceptable

in the event of failure.

C. Latency

When latency is introduced in electronic musical instruments it can prevent performers from playing consistent rhythms. Delayed audio feedback is a huge 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 100ms of latency, the average performer can only compensate for around 30ms of latency [2]. Therefore our system should hit this 30ms target in order to be accessible to the widest range of users. Additionally, contact detection must be extremely responsive, and as such the sensing gloves must be able to correctly identify contact with an object 99% of the time at minimum.

D. Motion Tracking

The sensing glove will be responsible for detecting movements of objects. Ultimately the user wants to feel that their actions are consistently changing the sound, so absolute positional accuracy is not required. Therefore our system must be able to stably track the glove's motion with a success rate of 80%.

E. 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.

F. Ease-of-Use

Finally 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 tutorialized GUI. This must be rendered fast enough so that the user does not experience a noticeable delay or become nauseous. The target framerate is therefore 30fps.

III. Architecture and/or Principle of Operation

Our system architecture consists of three distinct components. Each component 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 possibly being touched by the user. The CV component consists of a headset with a mounted webcam (for video input) and a display (for video drawover output) as shown in Fig. 1. For our MVP, we will only have a headset-mounted webcam, and the display will be an external monitor. The headset is powered by and communicates with the NVIDIA Jetson Xavier AGX 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 continuously sent to the MIDI processing component to determine which type of signal should be output. We also perform video drawover and display the output to the display to provide real-time feedback about what the system is doing.

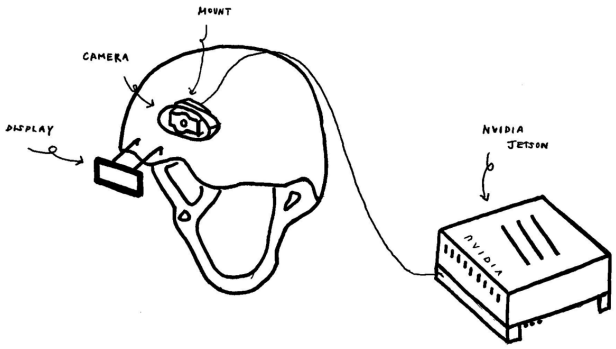

Fig. 1. A sketch of the headset with a mounted webcam and display

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 moves that object in space along 2 translational dimensions (left-right, up-down). 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, which processes the raw sensor data into a packet containing a contact-detection flag and two-dimensional coordinates of the glove's estimated position. This conversion is done with algorithms based on physics and signal processing, which we will implement. The output packet is sent to the MIDI processing component to determine the corresponding MIDI signal parameter values.

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 maps these values to the appropriate MIDI signal parameters and acts as a USB MIDI output controller.

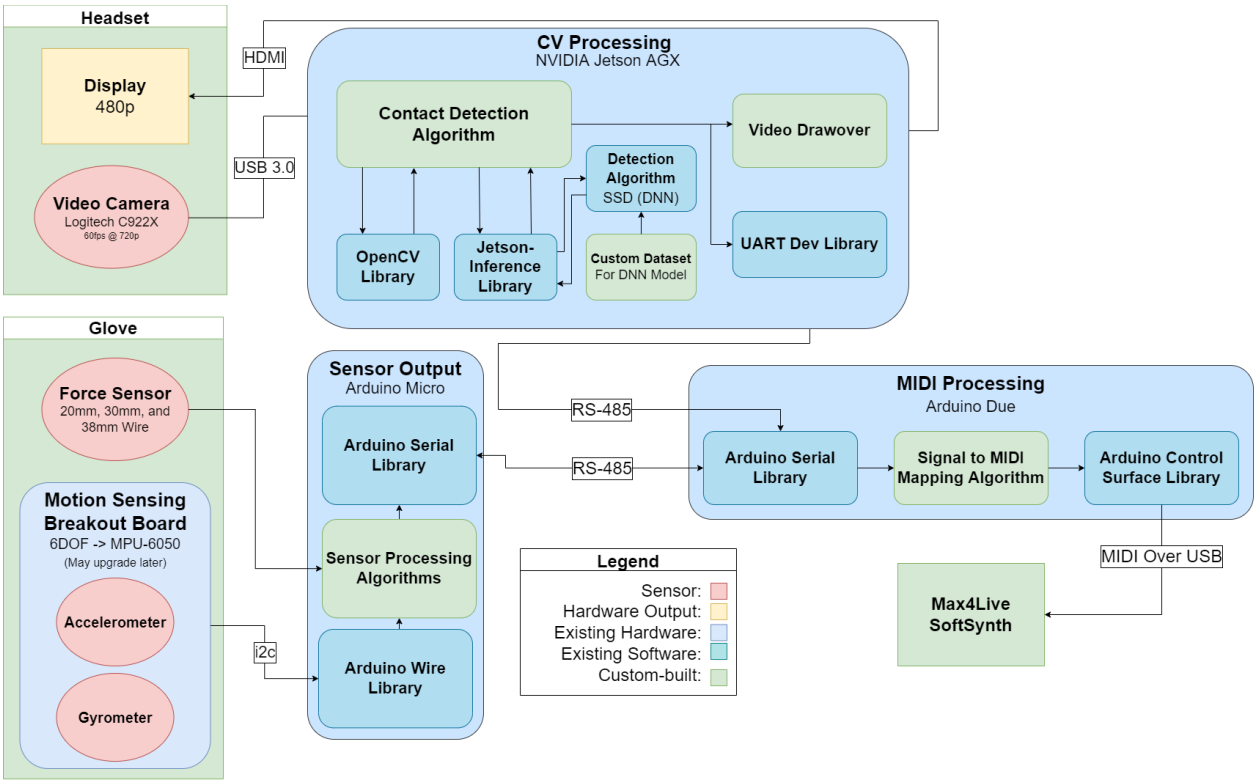

Fig. 2. Full-scale system block diagram

In order to showcase the functionality of our project the MIDI output data will drive a custom-built Max4Live software synthesizer. Max4Live is a graphical programming language built for processing audiovisual signals and running them directly into the Ableton Digital Audio Workstation as shown in Fig. 2. Building a software synthesizer with custom MIDI mappings is trivial with Max4Live, and is the perfect way to showcase the capabilities of our controller.

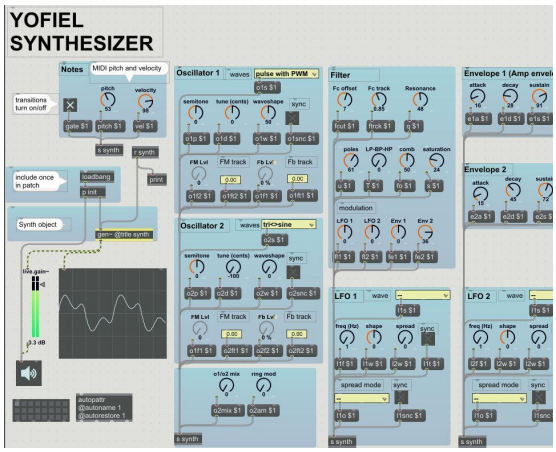

Fig. 3. An example of a Max4Live device. The Yofiel Synthesizer [3], made entirely in Max

IV. Design Requirements

As stated above, the system must support a reasonable note granularity such that the user can play complex melodies and rhythms. At 200bpm a whole note can be played every 0.3 seconds, and thus a 1/16th note can be played every 0.018 seconds. Similarly, a 1/8th note can be played every 0.037 seconds. This means that the camera must capture video at 60fps to detect 1/16th notes and 30fps to detect 1/8th notes.

$$\frac{60\ Seconds\ per\ Minute}{200\ Beats\ per\ Minute} = 0.3\ Seconds\ per\ Beat$$

$$0.3\ Seconds\ *\ 1/16th\ Note = 0.018\ Seconds$$

The computer vision detection must also be able to operate at these speeds as well. If the model is unable to perform detection at this rate, then interpolation will be utilized to simulate the same performance using predictive techniques.

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. It is expected that computer vision detection and board-to-board communication will 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 11ms. In order to give us some slack room we are therefore establishing a 15 ms latency requirement for the CV portion of our system. The performance capabilities of the Jetson combined with clever optimizations of the model used should allow this target to be reached. The

communication protocol of the system must perform at a high speed as well. Given how little processing should be done for this, a target of 5% maximum latency for communication has been established. This works out to a 1.5ms maximum. The current approach we are planning to use, RS485, can perform operations at around 3Mbps. Given how little data each packet contains this should be more than adequate for satisfying this requirement. 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 book, a cup, and an apple. 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. Given that the output of this sensor is a raw analog voltage, measurements only need to be taken as fast as the note granularity dictates, so every 0.018 seconds at minimum. The gyroscope broadcasts its output over RS485, and as such its data will be broadcast at a rate of up to 10 Mbps. It must be able to extrapolate a new X and Y position accurately enough to meet the 80% threshold, and this will be measured through successes over various motion trials.

The video streaming and drawover must occur at a framerate of 30fps at minimum to ensure a smooth viewing experience for the user. The GUI must be drawn over the video, and while it would be ideal to do this at the framerate of 30fps ultimately this is not necessary if it contributes to latency. Therefore an absolute minimum of 30fps for the GUI drawover is required.

V. Design Trade Studies

Our project consists of multiple subsystems, each requiring multiple design choices and tradeoffs. Here we have detailed our thinking behind 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 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 an Arduino Due to run MIDI over USB. Most musicians who are new to electronic music production will be using a computer to produce, so Software MIDI was chosen to maximize accessibility. As stated in (2.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. If we have more time we are absolutely interested in adding hardware MIDI support.

B. Communication Protocol

Our system consists of three computers all trying to communicate with each other over a full-duplex system. As such we needed to decide on a communication protocol. We ended up choosing between three options: I2C, RS485, and Bluetooth.

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, I2Cs 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.

The next choice is RS485. This communication protocol is implemented through differential voltage levels, and is generally used in high-noise applications. It is very stable even at our operating voltage of 3.3-5V, and is rated for distances of up to 800m in some applications. It can achieve transfer speeds of up to 10Mbps, 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 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 huge advantage of being wireless, and would allow the user to untether the headset from the glove. Bluetooth can run at speeds of up to 3Mbps depending on which architecture version is used. It also has an effective range of about 300ft. However, Bluetooth also does not handle arbitration particularly well, and is a lot more complicated to implement. We would have to consider dropped packets a lot more, 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 RS485. 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. With regards to Bluetooth, we felt that the added complexity of supporting a wireless system and the issue of arbitration for multi-directional communication were too concerning 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. RS485 is fast, stable, and long-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 kpbs (Fast Mode)	8 in.	Yes
RS485	10 Mbps	2600 ft	No
Bluetooth	3 Mbps	300 ft	No

C. Complementary vs Kalman Filter

MEMS sensors generally are designed to be cheap, and thus tend to introduce erroneous drift over time. Accelerometers are designed to measure the acceleration with respect to the earth's horizon axis but they will show zero acceleration for an object in freefall. Given that accelerometers alone cannot account for 3D rotation a gyroscope must also be utilized. However, 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 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 remove accelerometer spikes and Gyroscopic drift relatively. The Kalman filter is an iterative filter, which is efficient but has high computational complexity. It works by correlating between current and predicted states. The advantage of Kalman filters is that they can run on devices with very small amounts of memory. All of the forces working on the object are measured by the accelerometer, and as the small forces create disturbances in measurement the long term measurement becomes less reliable. 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, so a high pass filter is needed for gyroscopic data correction. The complementary filter consists of both a low and high pass filter and as it is easier to implement this filter was implemented for getting precise data. Fig. 3. is the block diagram of the complementary filter.

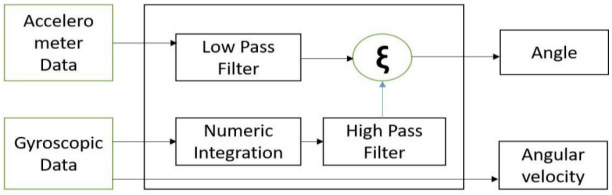

Fig. 4. Block diagram of complementary filter

For a dynamic system the current state can be inferred by using previous data to predict future values. For this type of system which is continuously changing the Kalman filter is the perfect choice to make an educated guess of what happened. The Kalman filter uses correlation between prediction and what actually happened to estimate the prediction error. The procedure is divided into four steps as shown in Fig. 4. Firstly the initial value is given, then the prediction step, then the gain of the filter is computed, and finally the estimation is done.

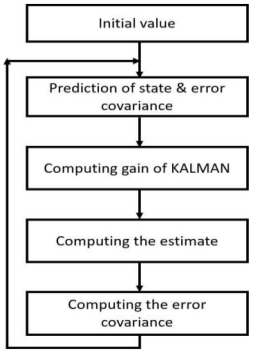

Fig. 5. Kalman filter flow chart

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 our project as we may implement two gloves in the future. 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 and this requirement doesn't fit our decision of using an Arduino Micro, which is a 8 bit MCU. Therefore, we will be using Complementary filters over Kalman filters.

D. 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, we determined that robustness and reliability is the most important metric for detecting and tracking objects. With the NVIDIA 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. A jar of mayonnaise, the namesake of our project, is not included among this set. 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]. Ultimately, this approach can produce a model trained on fewer images, within a shorter amount of time, while retaining the accuracy of the parent model. We can 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 will be augmented with the sensors and processing boards that we will be attaching to it. Therefore, we would likely need to retrain the model on a custom dataset including images of the finalized glove design. This would introduce 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 (see 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 labeled with them. OpenCV provides an implementation for detecting these markers in an image, which we have tested to run in real-time [10].

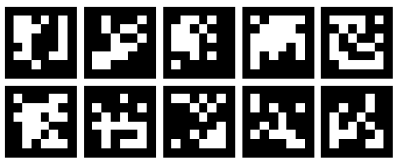

Fig. 6. A set of 10 arUco makers [10]

VI. System Implementation

As stated earlier, our project will be 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 diagram of the system components and interconnects is shown in Fig. 6. The details of the various subsystems within the project are detailed below.

A. MIDI Processing

All incoming sensor packets will be transmitted to an Arduino Due. The Due shall utilize the Control Surface library to act as a native MIDI USB device. It will then utilize these libraries to map the incoming sensor data to MIDI signals. This will result in four distinct parameters for our MVP: Contact detection (MIDI note on or off), Object identification (What waveform(s) is/are being used to drive the synthesizer), and the X and Y coordinates of the object (General analog parameters). In order to perform this mapping the Jetson shall transmit packets containing a numeric ID corresponding to the object class it believes is currently being touched. This will then be compared with the sensor data to determine if contact is actually happening. If so, a Note On MIDI signal will be sent, and when contact ends a Note Off MIDI signal will be sent. Additionally, along with these signals, a MIDI value from 0 to 127 will be sent using a General Purpose Controller signal which is mapped from the ID of the object detected. A new MIDI signal for both the X and Y coordinates will only be generated while contact is occurring. Both the X and Y coordinates will also be mapped to General Purpose Controller signals. Last recorded X and Y coordinates will be stored for each object ID the Due receives. 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.

TABLE 2. MIDI SIGNAL MAPPING

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)

B. Video Display

The video input from the camera will be drawn over on the Jetson. Currently the plan is to use Gstreamer and Qt to do video drawover directly on the Jetson and stream the output over HDMI to an external 480p display at 30fps. Notably, the video feed FPS does not need to match the camera's FPS to have smooth performance. If possible, the Arduino Due will also send sensor data information to the Jetson so that it can visualize this data on the display. For our MVP this display will simply rest on a tabletop, but if we have time we will mount the display to the headset.

C. Board-To-Board Communication

The Jetson, Micro, and Due will all communicate via full duplex RS485. All of these boards support TTL over UART communication. Cheap and affordable TTL-to-RS485 chips like the Max 485 exist and are deployed in industry all the time, so we will be utilizing a set of these to handle the board-to-board communication. The Arduino Due shall act as the Controller, and use its multiple serial ports to facilitate communication with the Micro and the Jetson. The Due will request sensor packets from the Micro and computer vision output packets from the Jetson. The Due will also transmit the sensor packet data along with the request to the Jetson so that the data can be displayed to the GUI Display. As stated in (5b) the RS485 protocol does not support hardware arbitration, and as such we will have to do all of our arbitration in software on the Due.

Unfortunately the Micro and the Due do not have the same operating voltage (5V vs 3.3V). Therefore when building the hardware for this interface we will need to do some logic level shifting at some point in the interface. We have two options for Max 485 boards used to facilitate the TTL-to-RS485 interface. One option claims to have built-in logic level shifters, but we are unsure as to how reliable this system is so we will have to test it before using it. If it does not work, we have hardware logic level shifters ready to be deployed between the Micro and the Max 485.

D. Computer Vision

The CV subsystem provides *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 (helmet) mounted with a webcam and display, as well as the NVIDIA Jetson Xavier AGX, an AI-accelerated CV processing board as shown in Fig. 1.

For video processing, we will extensively use the *Jetson-Inference* library, which is a real-time inference library for deep neural network applications developed by NVIDIA for deployment on Jetson boards. We chose this library because of its optimization and ease of integration with our specific hardware. We may also rely on the *OpenCV* library for computer vision processing tasks.

Real-time object detection is performed using the *SSD-MobileNet* deep neural network, which was designed for embedded computing applications and initially appears to meet our latency requirements. For our DNN model, we will retrain the *SSD-MobileNet* model that is provided by *Jetson-Inference* on a custom dataset of images containing our chosen set of objects to identify. This dataset will be composed of existing, labeled images from the Open Images dataset of mugs and pens, as well as images of mayonnaise jars that we will capture and label ourselves. We aim to have at least 100 images of each item.

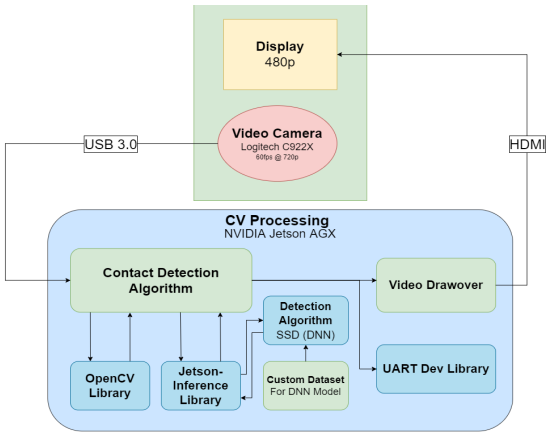

Fig. 7. A block diagram detailing the workings of the Computer Vision hardware and software.

Potential contact detection will be implemented by tracking the glove and identifying its distance to the detected objects in the frame. The glove will *not* be tracked using the neural network for object detection, for the reasons described in Section V. Instead, we will be tracking the glove with an algorithmic computer vision approach by using fiducial markers attached to the glove. Specifically, we will use ArUco markers, which are NxN binary images that act as reference points for augmented reality applications. The detection of these markers will include deriving each marker's type and corner locations in the image. By attaching multiple of these markers in different locations on the glove (see Fig. 8), we expect that at least one marker will always be visible in the camera's view, from which we can derive a bounding box for the glove using graphical transforms.

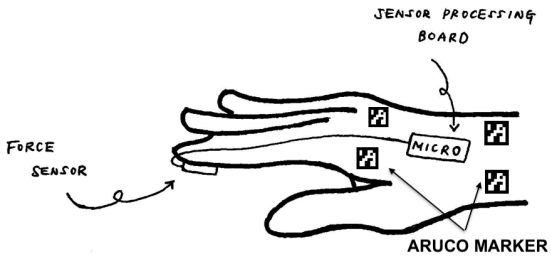

Fig. 8. arUco markers printed and attached to the glove at multiple locations.

Potential contact detection with detected objects will then be determined by checking for overlap between the bounding boxes for the detected objects and the tracked glove. If multiple objects overlap with the glove, then the tie will be broken by minimizing the distance between the object and glove bounding box centers.

E.. Motion Sensing

The motion sensing subsystem delivers users direct interaction with daily objects by assessing their position and movement. The goal of this component is to examine whether a user is touching an object or not, and detect any translational or 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 will use a fingertip force sensor, accelerometer, gyroscope, and an Arduino Micro. As shown in Fig. 8. all the sensors will be connected to the Micro and the board will be placed on the back of the glove. The board will be placed in a plastic casing that will protect the Micro as shown in Fig. 9.. This clear plastic case will be attached to the back of the glove using epoxy or hot glue.

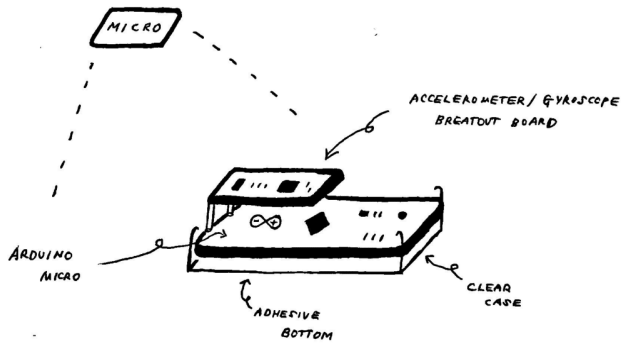

Fig. 9. Arduino Micro and its clear plastic case

When implementing a force sensor in circuitry as shown in Fig. 10, 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. 11. According to the graph, we plan to use a resistor with a value of 10K to achieve the widest range of sensitivity.

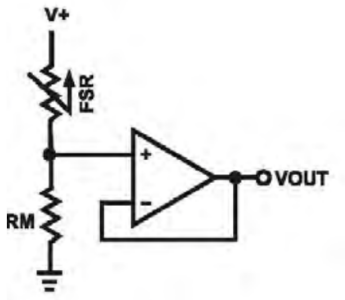

Fig. 10. A simple circuitry implementing a force sensor. Retrieved from [12]

Initially, the force sensor will output 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 will consider a threshold voltage of 2.64V. This value is 80% of the supply voltage and we are confident that the value will let us know of the concrete contact with an object. Once the threshold voltage is acquired on the Micro, the board will convert the data into a binary representation of 0 representing no contact, and 1 representing concrete contact. We don't expect 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.

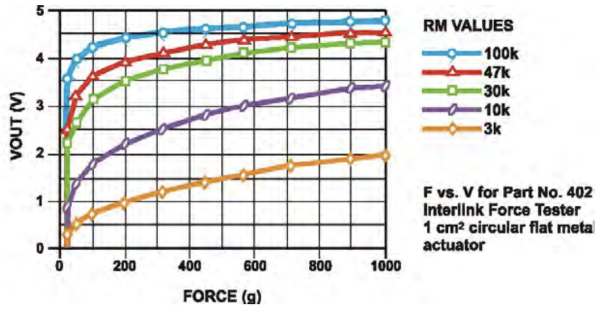

Fig. 11. The plot of sensitivity-based on five measuring resistor values. Retrieved from [12].

Raw data obtained from the gyroscope and accelerometer need to be post-processed for them to be meaningful. Since the accuracy of these parameters are influenced by a number of errors which are a function of time, we will first identify the errors in the sensors and take a signal processing approach to minimize the errors.

The gyroscope drift is mainly due to 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 can be removed within a gyroscope through the use of accelerometer feedback to monitor position relative to gravity. Filtering the gyroscope output using a low-pass filter will cancel a portion of the drift error. The cutoff frequency of the low pass filter will be determined by taking the average of output signals and checking its frequency response. Also, we will 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 can be nulled to zero for that respective axis. This will allow the sensor to correct for the bias instability. The clear representation of the whole process is shown in Fig. 4.

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 will be required to have a calibration time of approximately 15 seconds to capture bias occurring when the glove is supposedly in a stable/rest state. This time will allow 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 are expecting to use a low-pass and a high-pass filter. The acceleration measurement will pass through the low-pass filter to remove 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. Using the acceleration data, we will need to integrate over time twice in order to acquire positional data. During this process, we will high pass filter the velocity signal to yield an integrated signal. Here, the high-frequency drift error (VRE) is the response of an accelerometer to current rectification in the sensor, causing a shift in the offset of the accelerometer. Minimizing the problem is significant as the error propagates with time and can lead to over compensation in stabilization.

We will be implementing 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, position, and rotational data will be 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. MIDI Mapping

MIDI mapping will be tested using dummy input data during unit testing. Success is measured by the successful transmission of MIDI signals from the Due to the host computer over USB. MIDI mapping latency will be measured from the moment of data acquisition (dummy RS485 data during unit testing, actual RS485 data from Jetson and Micro in integration). If we find that it takes more than 1.5ms for this stage of the process this constitutes failure. In this event we will consider utilizing a hardware MIDI interface driven by the Due, or writing custom software to run on the Due.

B. Video Display

Video capabilities will be tested by attempting to draw over dummy video input from the Jetson and routing the output to a display. The FPS will be timed in software and compared against our requirement. If the target of 30fps cannot be achieved the drawover framerate of the GUI will be lowered. If this is also not sufficient then a hardware rendering approach off of the Jetson will be considered.

C. Board-To-Board Communication

The success of our communication protocol ultimately depends on the latency. For unit testing a set of Arduino Unos will generate dummy output data that will be sent over RS485. A software system intended to replicate the final integrated system will be deployed. For both the unit test and the final integration test the latency between generation of RS485 signals and unpacking of RS485 data will be recorded. If this does not hit our 1.5ms target for all communication directions this will constitute a failure. Given that RS485 is rated for up to 10Mbps, if we cannot hit this target we can only infer that the hardware being used for the TTL-to-RS485 is our bottleneck, and we will deploy replacement hardware as necessary.

D. Object Detection

We will test the accuracy of object detection by running the program in a series of controlled experiments. These experiments will focus on testing the robustness of the detection in relation to the motion of the headset and camera, the number of objects in the frame, the types of objects in the frame, and the obstruction of the object's view. We will run each experiment 5 times over a period of 60 seconds. We will count the number of frames in which any of the test objects were not detected when they should have been over the total number of frames for that trial. This will give us an object detection accuracy measure. To satisfy our use-case requirements, we expect the system to track at least 90% of the frames correctly in the simple case that the hand does not fully obstruct the object. The corresponding testing configuration is summarized in Table 3.

Table 3. Computer Vision Trials

Environment Parameter	***Trials***	***Trial Length***
Steady Camera	5	60 seconds
Moving Camera	5	60 seconds
1 Object in Frame	5	60 seconds
2 Objects in Frame	5	60 seconds
3 Objects in Frame	5	60 seconds
Partial Obstruction	5	60 seconds
Full Obstruction	5	60 seconds

E. Potential Contact Detection

The testing procedure for potential contact detection will share the same trial configurations as the object detection tests. In each trial, the user will be repeatedly instructed to reach for any of the objects in view. The time at which the user touches an object will be recorded, along with the type of object touched. The corresponding contact prediction will be recorded, and later cross referenced with the ground truth. The evaluation metric will be the percentage of times that the correct object type is predicted at the time the user reaches for it. We consider a passing rate to be 90% accuracy in the same simple case described above.

F. Motion Sensing

We will test for five measures to determine the success of motion sensing in our system. First, we will check for force sensor functionality. There will be three objects required for this test: a mug, a pen, and a jar of mayonnaise. Using these objects, we will check the binary result of the sensor's touch recognition.

Second, we will examine force sensor sensitivity. Again, we will use the same three objects used for testing the functionality of the sensor. Since the sense of touch is very subjective, we have defined four different pressures to test: the pressure of touching a baby (less than 200fg), grabbing a mobile phone (200-500fg), gripping a baseball (500-800fg), and strongly grabbing on to a weight (800-1000fg). Based on the subjective results acquired by each team member, we plan to take an average of the sensitivity measure acquired from five trials. Then, we will find the difference between the average and the center value of each force gram range. If the average of differences is less than 200fg, we will choose the according sensor for the board. This method of recording each

member's fingertip force and comparing with expected metrics will allow us to find the suitable force sensor.

Third, we will inspect for accelerometer sensitivity. When starting the testing procedure, each member will place their right hand in front of one's face with the palm facing down to calibrate and set a starting point. Once calibration is completed, one will move the hand to the left then to the right as much as possible. In a similar fashion, one will also move the hand in the up direction and then the down direction as much as possible. Considering that most accelerometers range from ±1g up to ±250g, we will check if all the test processes reach ±125g. If the success rate is above 80% for all directions, we will consider that this test is passed.

Fourth, we will look over for gyroscope sensitivity. Again, each member will calibrate using the same method as mentioned above. Then, one will rotate the hand to the left then right as much as possible. A full range of 450deg/s is sufficient for most industrial applications, such as maritime applications, satellite stabilization, camera stabilization and aerospace applications. Our current breakout board's gyroscope range is ± 250 500 1000 2000°/s. And so, if the average of the range covered by three team members is above 80% accuracy when compared to 450 deg/s, then we will consider that this test is passed.

Lastly, we will examine ergonomics and usability of the glove. We expect that the wire connecting the force sensor and the Arduino may not provide a room for a person to move their fingers freel. In order to test for any discomfort, we will have a series of testers grab and let go of a water bottle 50 times using gloves that have different wire length configurations. We will record each person's subjective opinion on their experience (rating of 1-10). If the average score is above 7, we will consider that this test is passed.

VIII. Project Management

A. Schedule

Our schedule is divided into five sections: Project Logistics, CV Detection Implementation, Sensor Board Implementation, MIDI Board Implementation, and Video Board Implementation. Fig. 13. is the Gantt chart that the team is following and also reflects the five sections. Notice some task descriptions have 'T-' and this means that task is related to testing of a specific component in the system. For all the sections, we expect to acquire and process data in each individual board before the start of Spring break. After the break, the team plans to focus on communicating those processed data between the boards and integrate them into a system. If all go as planned, we hope to add more features after the Interim Demo and perform extensive usability tests.

B. Team Member Responsibilities

Harry Fernandez will be working on MIDI output mapping and its signal generation as well as the board-to-board communication protocol. Min Gun has the responsibility of implementing the physical sensor processing board. If it is necessary to PCB design our sensor board, he will be in charge of the design as well. Min Gun has a secondary job to create a Max4Live software synthesizer that can work well with the system for any demo showcases. Tomas will be working on CV algorithm implementation on the Jetson. Along with implementation and testing of the algorithm, he will train models for object detection. Furthermore, he will work on the design of visual overlay and the implementation of the output video display.

C. Bill of Materials and Budget

Table 1. is the bill of materials that keeps track of all the parts that we have purchased so far. We are certain that our future total cost will be far below the limitation of $600.

Table 4. 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	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

Item Description	*Model Number*	*Manufacturer*	*Qty.*	*Cost*
Jetson Xavier AGX	N/A	NVIDIA	1	Inventory

D. Risk Mitigation Plans

According to our schedule, the biggest concern that we have is not receiving the parts that we need on time. In particular, the forceWe sensors, accelerometer, and gyroscope are heavily dependent on the manufacturers' shipping time. In order to mitigate this risk, we are planning to set up all the groundworks before the sensor arrives. When they do arrive, we hope that already developed code can be used for testing just by plugging the sensor into the microcontroller.

Another major risk that we foresee is meeting our use-case requirement on latency of the system. We have set that we want to meet less than 30 milliseconds of end-to-end latency across the system. Even though the Jetson board meets the technical specifications, we believe that there can be issues when developing a network with multiple processing boards. We hope to overcome this possible risk by optimizing communication between boards over RS485 and refining sensor data as much as possible. Moreover, there is a possibility of combining a few functionalities into one board when we are sure each modular component is working correctly.

IX. 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.

X. Summary

We intend to create 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. This will enable more musicians to quickly jump into electronic music production, while also providing a tactile and experimental experience for seasoned producers. The biggest challenges we hope to overcome are the latency and accuracy of our system, but we are confident that the hardware and software approaches that we have planned will enable us to satisfy these requirements.

Glossary of Acronyms

AR - Augmented Reality
FPS – Frames per second
CV – Computer vision
I2C - Inter-Integrated Circuit
Jetson – NVIDIA Jetson Xavier AGX
MIDI – Musical Instrument Digital Interface
SSD - Single Shot Detector
USB - Universal serial bus
VR - Virtual reality
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] Ernest. (2016, December 13). Yofiel All-in-One gen~synth - maxmsp forum: Cycling '74. Cycling74 Forums. Retrieved from https://cycling74.com/forums/yofiel-all-in-one-gensynth

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

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

[9] [F] Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., & 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] [G] OpenCV, (2022). The OpenCV library website: https://opencv.org

[11] 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

[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

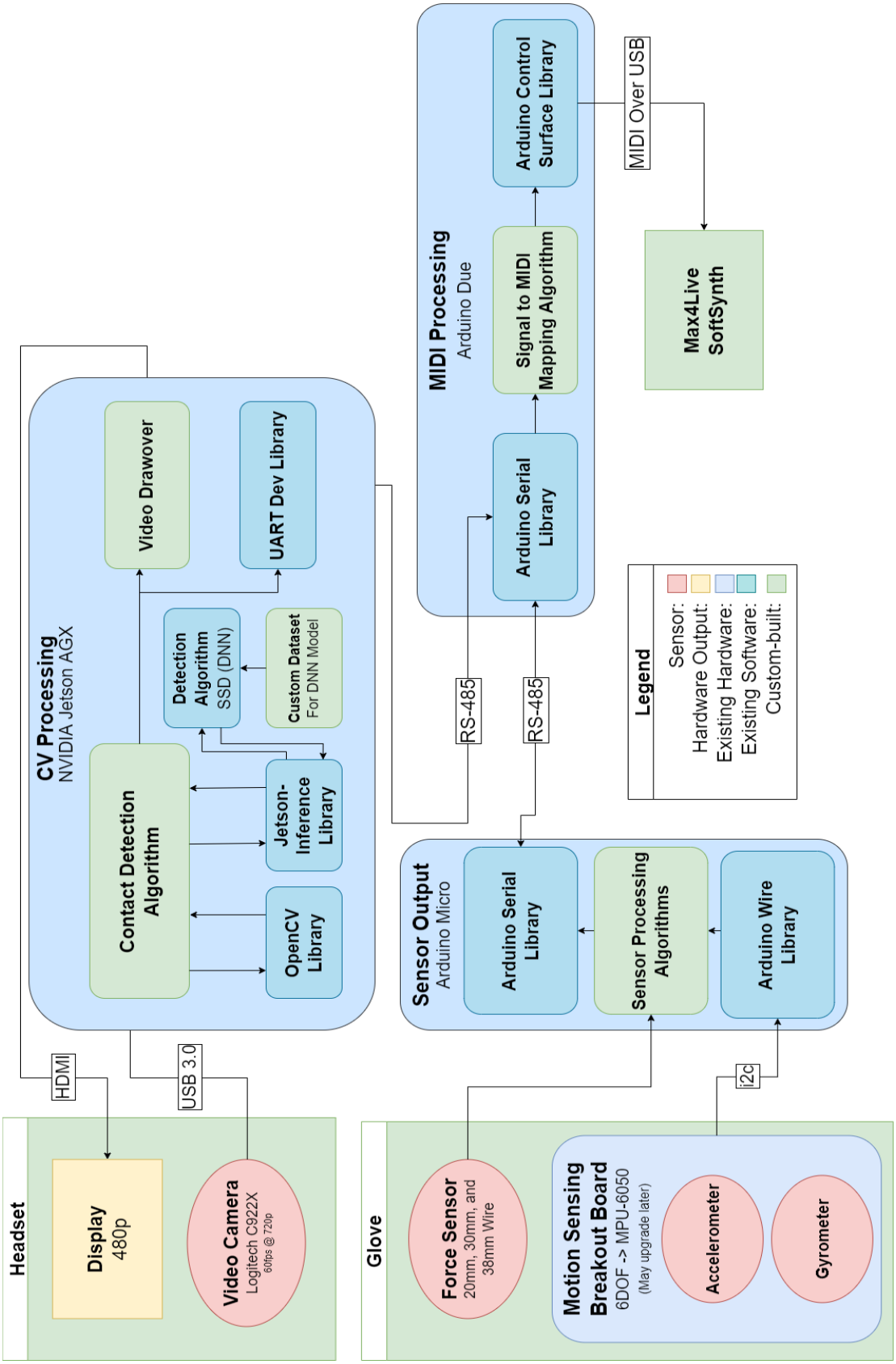

Fig. 12. Full-scale system block diagram

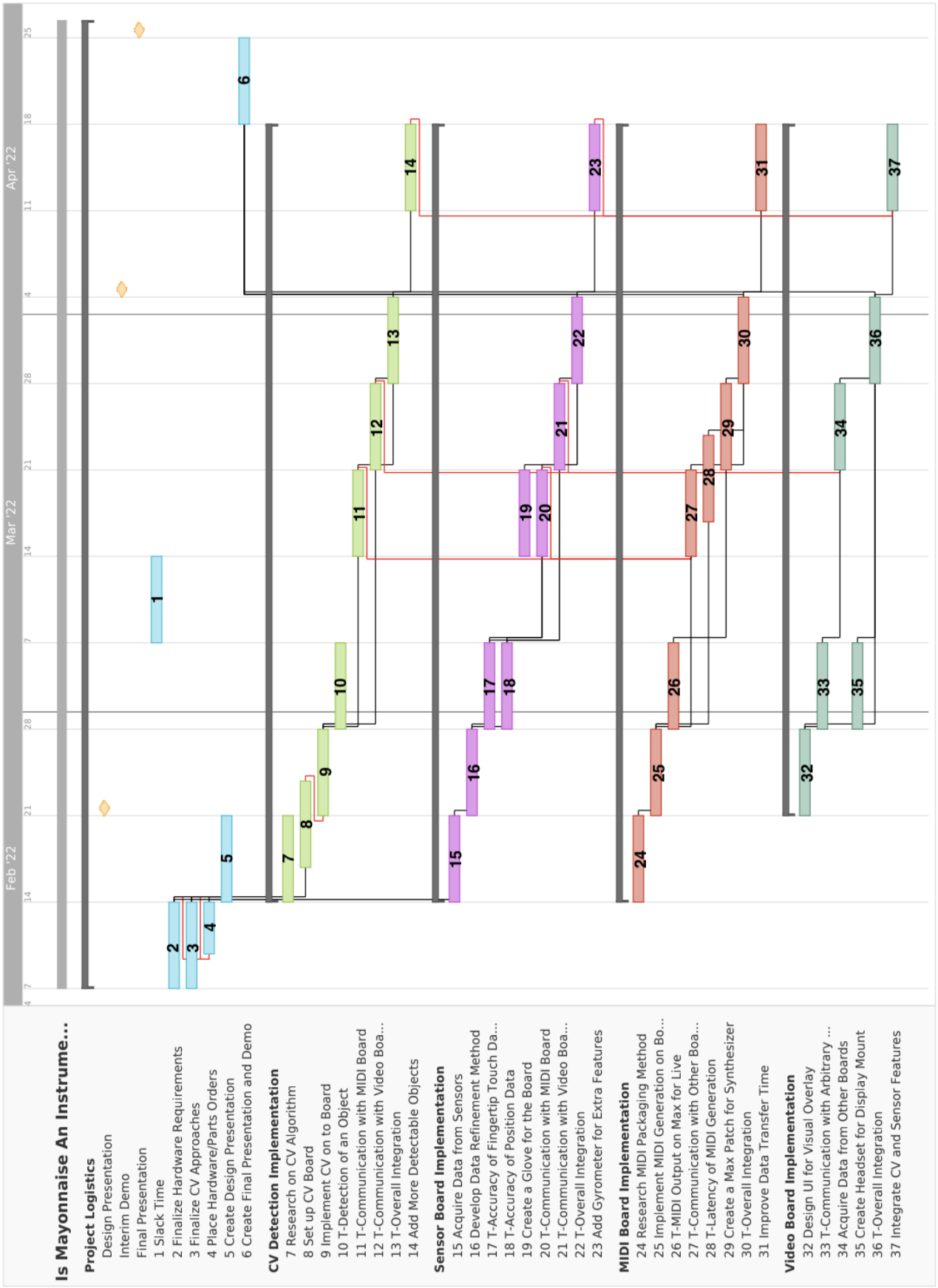

Fig. 13. Schedule example with milestones and team responsibilities