Memorandum

To: Kimberly Lemieux
CC: Capstone Instructors
From: Wes Marsen, Andrew Goertzen, Bradley Walker
Date: September 19, 2025
Subject: WAW MIDI Controller Proposal

Summary

Have you ever wanted more out of a MIDI controller? Have you noticed that almost all MIDI controllers look the same? Our team is breaking the mold on conventional controller styles with a new, innovative design; rather than traditional keys and knobs, we will use capacitive touch sensors as inputs, including keys, modulation sliders, and an X-Y pad. We will use a USB-C connection for power and data transfer to computers, eliminating the need for a dedicated power supply, and a MIDI TRS jack to connect to other MIDI capable devices. As of week-three, we have spent over $100, and we intend to keep our development costs below $500. Our goal is to sell our controller, the MTP-25K, as a user-buildable kit costing less than $200. It will be industry ready for the capstone presentation on December 12, 2025.

Introduction

Musical Instrument Digital Interface (MIDI) controllers are devices that convert gestures and tactile inputs into musical information, allowing musicians to control both virtual and physical instruments. Most controllers follow a familiar layout: piano-like keys at the bottom and an array of knobs, pads, or sliders above [1]. While this design is effective, it has become the industry standard, leaving little room for fresh approaches to performance and creativity.

Figure 1: example of a lower to mid-range MIDI controller: AKAI MPK mini [2].

The market is currently saturated with traditional MIDI controllers, ranging in price and features. Budget controllers under $100 typically exclude keyboards but include a few knobs and sliders, while mid-range models, such as Figure 1, ($100–$500) combine limited keyboards with pads and dials. High-end controllers, priced above $500, add more pads,

sliders, and advanced features such as sequencing and arranging. This creates a gap where musicians must choose between affordability and functionality.

Our project aims to bridge this gap by designing a MIDI controller that is cost-effective yet comfortable. Instead of traditional piano keys, our design will feature capacitive touch keys and offer a different look and feel that encourages new play styles. By combining durable, affordable components, we hope to provide musicians with a tool that sparks creativity beyond the standard controller format.

Technical Overview

We have put a lot of thought into the design and specifications of our controller. Meeting class and industry standards, such as the General MIDI standards set by GENERAL MIDI in figure 2, will be our overarching mission. Reading this document will reassure you that this project is well planned, thought out and worth backing.

Figure 2: General MIDI logo [3]

Hardware

The primary purpose of this device is to provide an intuitive and responsive physical interface for playing music that is portable and robust enough to be transported in a bag or laptop case. To achieve this, we will combine a recognizable keyboard-style layout with non-traditional PCB-integrated capacitive touch surfaces, like Figure 3, for the playable controls. This approach avoids the mechanical complexity of a physical keybed, knobs, and sliders while retaining the sense of tactile feedback through both physical cues (the touch surfaces are physically recessed into the enclosure, so the player can feel them) as well as visual feedback via low-power LED indicators. The keyboard will also provide velocity sensitivity through the use of piezoelectric contacts mounted internally to the keyboard area of the PCB, which will sense the magnitude of the physical disturbance of the unit by the player as notes are played [5].

Figure 3: Capacitive Touch sensors for breadboarding [4].

The device will be bus-powered and class-compliant over USB-C, meaning it will provide a simple “plug-and-play” experience for the user. All necessary information on the status of the device will be displayed on the front panel with indicator LEDs. The input surfaces will consist of three main areas:

1. A 25-note, 2 octave “keyboard” with velocity sensitivity, closest to the player.
2. 4 touch-sensitive sliders which will provide MIDI data in one dimension, in the upper left corner.
3. XY gesture pad that provides MIDI data in two dimensions, in the upper right corner.

The processor will use these control surfaces to generate and send note and continuous control (CC) data through the USB-C connection to a USB host computer, or through the MIDI Out 3.5mm TRS jack to another MIDI-enabled electronic instrument.
The 'brains' of our project is the Teensy 4.1 development board, which is based around the NXP iMXRT1062 System-on-Chip (SoC). This board provides all the necessary processing power, speed, inputs/outputs (I/O) and hardware to bring our design to life. In our final design, the NXP SoC and other necessary hardware components will be built directly onto our PCB.

Software

This project requires minimal latency to avoid delays between user input and resulting data. As such, we've chosen to use the C++ language for its low overhead. We are using some existing C++ libraries, primarily usbMIDI, to streamline development. We started building "proof of concept" code in the Arduino IDE and have now transitioned to the PlatformIO IDE for Visual Studio Code better version control through GitHub.

The code will read digital and analog inputs from touch sensitive components and determine the correct MIDI signals to send to a computer connected by USB or other MIDI devices through a TRS MIDI connection. Due to the number of inputs and LED indicators, we will use additional integrated circuit chips to extend the main processor's GPIO. These will communicate with the main processor through the I2C protocol.

The controller will also include some standard features: octave up and down buttons to change the note range of the keys; an arpeggiator button, which will play held down notes in an arpeggio rather than simultaneously, with timing selection (sixteenth notes, eight notes, or quarter notes); and a sustain button, which will continue playing the notes after the keys are released, with mode selection (first four notes sustained, FIFO four note sustained, or all notes sustained).

Facilities

At Camosun College we have access to oscilloscopes, multimeters, power supplies, function generators, soldering stations, 3D printers, and a pick-and-place machine for surface mounted PCB parts. We will reach out to Filip Pietruszewski at Capital City Transistor and Valve (CCTV) for general advice and breadboarding parts. We will outsource fabrication of the Printed Circuit Board (PCB) to JLBPCB and use the 3D printers at Camosun College to fabricate the enclosure.

Timeline

The project began before the start of the school semester; the group agreed on a concept in March 2024. The project will be complete for the Capstone Symposium on December 12, 2025.

Financial Overview

We will be funding the whole project except for the PCB, which will be covered by Camosun College. As of week three, we have spent over $100. We are confident we can keep the development cost below $500 since there won't be any physical dials or knobs. We have reached out to local music store CCTV to help with parts, and they donated several components for breadboarding.

Management Overview

Our team has been slowly forming over the last few years of school together. Last year we agreed that we would join forces and work together for capstone. It wasn't until the end of the winter semester that we started talking about a project idea.

Wes Marsen

Wes has spent over a decade playing music. He has been experimenting with MIDI controllers, electronic instruments and music production for 15 years now, and is the driving force for our project. Wes will oversee the design of the hardware as well as the Printed Circuit Board (PCB).

Andrew Goertzen

Andrew has always been into electronic music and has taken an interest in electronic devices. He is a father of two young children and is experienced in time management and conflict resolution. Andrew will be doing the written documentation of the project, the user manual and the website.

Walker Bradley

Walker has a history in IT and computer repairs. He has been coding for over a decade and since it is such a familiar language he will be tackling most of the coding. With such a strong foundation in C, C++ and C#, he is the best fit for our group to accomplish the goals on the software side. He also has 20 years of experience playing a variety of acoustic instruments.

Contract

To assure completion of the WAW MIDI Controller, all members of the team have signed a group contract outlining group goals, expectations, meeting times, decision making, conflict management, and consequences.

Conclusion

Our capstone project will produce a high quality and compact MIDI controller that will bring a fresh perspective to an often-generic product. It will inspire musicians to find a new way to play with a device they can immediately understand. We are confident that our product will be cost effective and different enough that it will appeal to many musicians. There is currently no design in the market like ours, the only ones that are similar are home-made controllers. One thing we've learned in our program is that breaking the mold and embracing change can be the key to success.

References

[1] What Is A MIDI Controller/Keyboard? What Is It Used for & Types – An in-depth write up about MIDI controllers.

[2] MPK Mini mk3 MIDI Controller | Akai Pro – Source for AKAI mini picture.

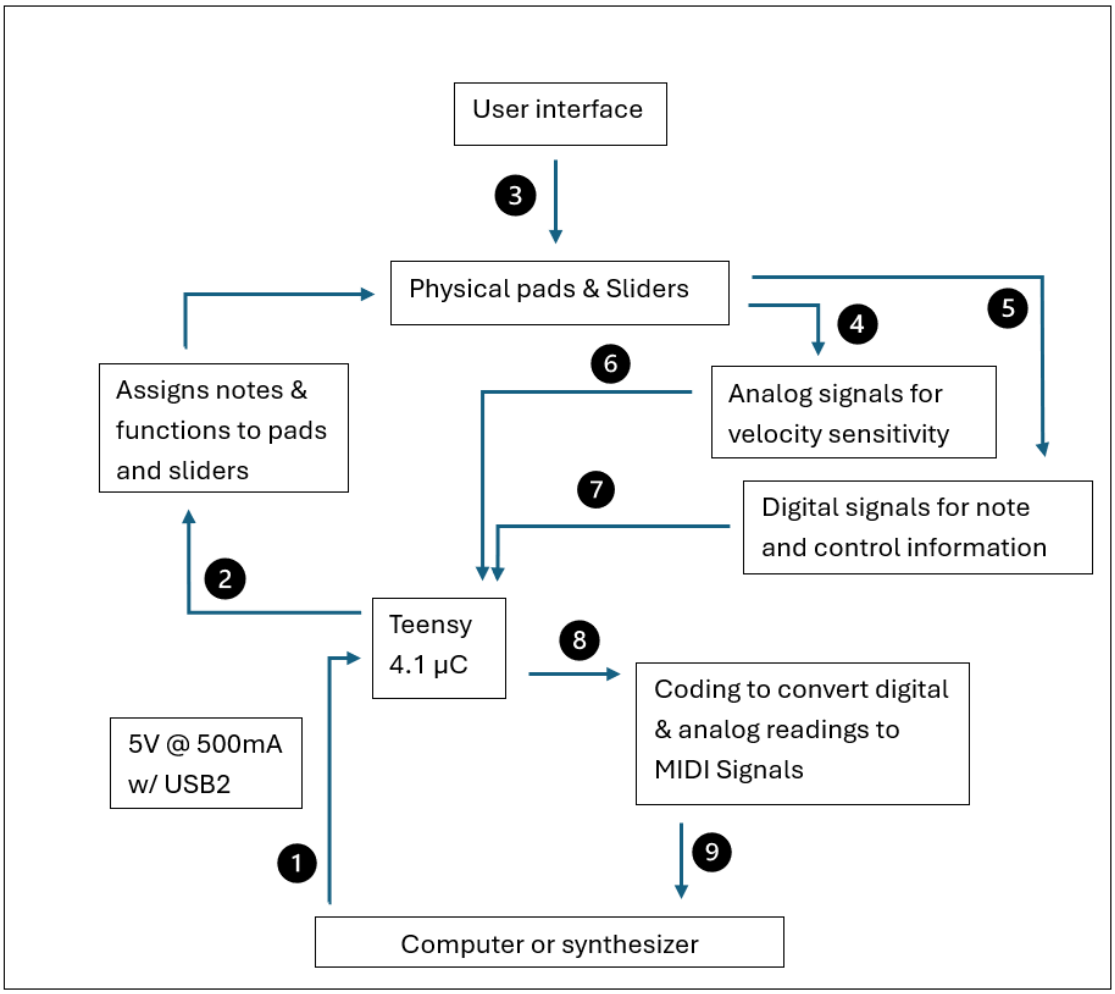
[3] LEARN PIANO using MIDI SONGS! / Dec 2019 / Report – General MIDI logo.

[4] Capacitive Touch Sensor TTP223B – Source for image of a touch sensor.

[5] Capacitive Touch Sensor Design Guide – Methods for creating a touch sensor.

Appendix A-1: System Block Diagram

Block Diagram showing how the user interface interacts with the computer/ synth.



Appendix A-2: Interfaces Between Blocks

Description of what is happening between each block in the block diagram.

Number:	Description:
1	Computer/synth supplying power the to the teensy board.
2	Teensy is using code to assign MIDI values to pads and sliders.
3	User is physically activating pads and sliders through touch.
4	Piezo sensors collecting analog values for how fast/ hard pads were pressed.
5	Touch sensors collecting *which* pad or slider is pressed.
6	Analog data of pressure is sent to the Teensy board.
7	Digital data of which pad/ slider is sent to the Teensy board.
8	Teensy board send all collected data to the software to be converted to MIDI data.
9	MIDI data is sent to computer/ synth.

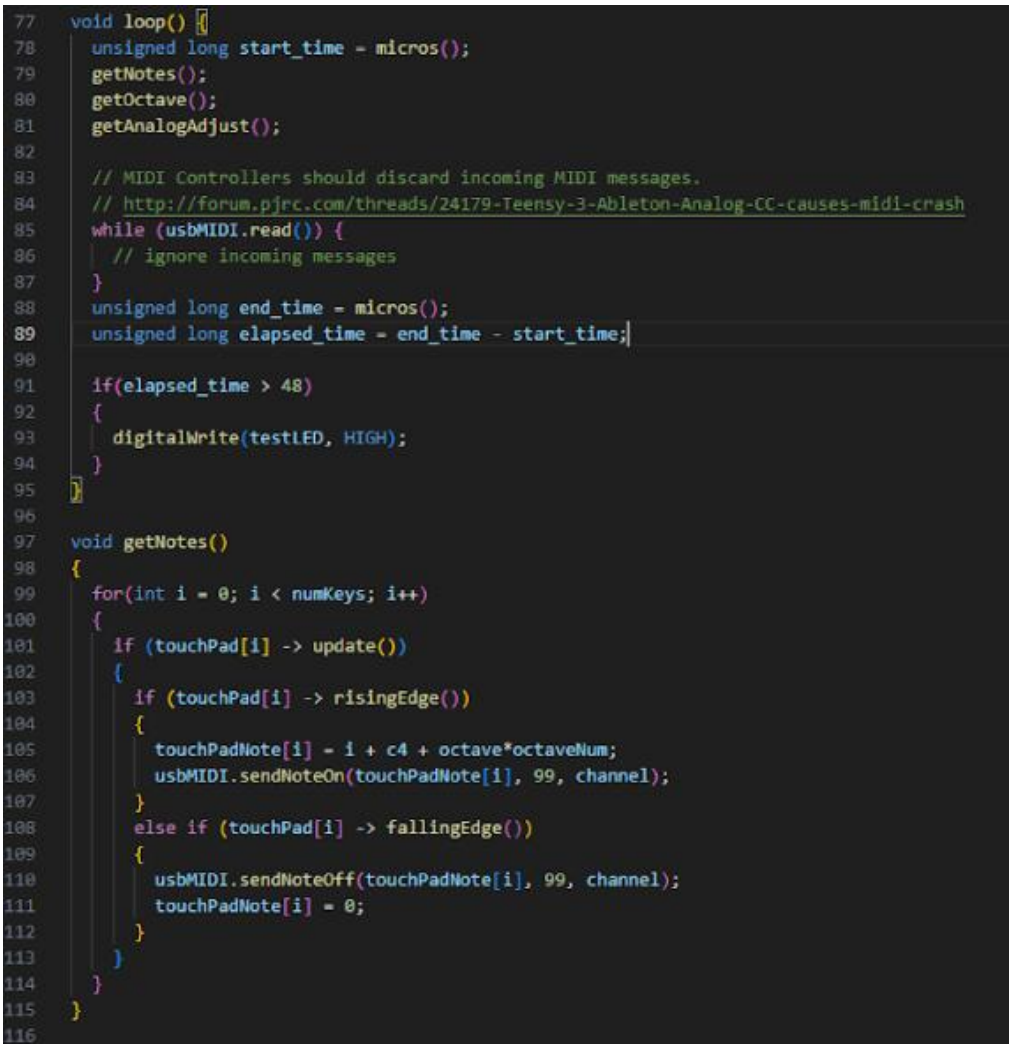
Appendix B: Requirements & Tests

A list of requirements for the WAW MIDI Controller.

Reference Number:	Requirement	Test:	Pass/Fail Criteria:
R01	Octave change buttons	Switches from one octave to next.	P- Octaves switch from one to the next.
R02	2 octave capacitive touch keys	Pressed and responds.	P- All keys work.
R03	Timing and speed = crisp	Measure the delay.	P – 1-10ms latency. <5 is the goal.
R04	Minimum of 2 assignable controls	Controls give appropriate feedback.	P – Slider works.
R05	USB connection for power	Powers the board.	P – powers all components at once.
R06	USB for data transfer	Transfer data to computer.	P – transfers all data successfully.
R07	External MIDI out Connection (TRS cable)	Sending MIDI data to external hardware.	P- Sends MIDI data to hardware.
R08	Communicates over MIDI protocol	Built and send data to computer.	P- sends MIDI data.
R09	Bug free code	Stress testing – try and break it.	F – unexpected behaviours and outcomes.
R10	Meets home/ consumer/ standards	7-year-old child test.	F - 7-year-old breaks it.

Appendix C: C++ MIDI Code

Here is a bit of our code we are using to implement the changes from digital signals to MIDI data.



```
void loop() {
  unsigned long start_time = micros();
  getNotes();
  getOctave();
  getAnalogAdjust();

  // MIDI Controllers should discard incoming MIDI messages.
  // http://forum.pjrc.com/threads/24179-Teensy-3-Ableton-Analog-CC-causes-midi-crash
  while (usbMIDI.read()) {
    // ignore incoming messages
  }
  unsigned long end_time = micros();
  unsigned long elapsed_time = end_time - start_time;

  if(elapsed_time > 48)
  {
    digitalWrite(testLED, HIGH);
  }
}

void getNotes()
{
  for(int i = 0; i < numKeys; i++)
  {
    if (touchPad[i] -> update())
    {
      if (touchPad[i] -> risingEdge())
      {
        touchPadNote[i] = i + c4 + octave*octaveNum;
        usbMIDI.sendNoteOn(touchPadNote[i], 99, channel);
      }
      else if (touchPad[i] -> fallingEdge())
      {
        usbMIDI.sendNoteOff(touchPadNote[i], 99, channel);
        touchPadNote[i] = 0;
      }
    }
  }
}
```

```
117
118  void turnOffNotes()
119  {
120    for(int i = 0; i < numKeys; i++)
121    {
122      if(touchPadNote[i])
123      {
124        usbMIDI.sendNoteOff(touchPadNote[i], 0, channel);
125        touchPadNote[i] = 0;
126      }
127    }
128  }
129
130  void getOctave()
131  {
132    if (octaveUpButton.update() && octaveUpButton.risingEdge() && octaveNum < 3)
133    {
134      octaveNum++;
135      turnOffNotes();
136    }
137    if (octaveDownButton.update() && octaveDownButton.risingEdge() && octaveNum > -3)
138    {
139      octaveNum--;
140      turnOffNotes();
141    }
142  }
143
144  void getAnalogAdjust()
145  {
146    analogReadResolution(14);
147    newPitchRead = analogRead(pitchAdjustPin);
148    if (newPitchRead > oldPitchRead+50 || newPitchRead < oldPitchRead-50)
149    {
150      usbMIDI.sendPitchBend(analogRead(pitchAdjustPin), channel);
151      oldPitchRead = newPitchRead;
152    }
153  }
```

Appendix D: Preview of Final Design