# 3 Project Status At End Of Phase I Work Period

## Timeline for Phase I

We anticipated this research and development (R&D) project to take 6 months from July 1, 2018–December 31, 2018. Many of the tasks were performed in parallel. The purpose of this first phase was to verify the technical feasiblity of creating this automated PT-treatment system. ***We have accomplished demonstrating technical feasibility.*** During Phase II of this proposal, we anticipate being able to do another prototype iteration of both hardware and software and prepare for clinicals. We will also improve the integration of movement and routines in the platform, expanding protocols and customization features as well as strengthen the patient assessment during warmup to deliver the best treatment for them at that moment in time. Based on our multi-faceted approach to find funding for continued development (grants, partnerships, traditional investors), we anticipate that we will be eligible for Phase IIb about 3-4 years from today.

## 3.1 System Architecture: "XELA"

In our early design, a Raspberry Pi 3 was going to serve as the primary controller of the system. It would communicate with the primary touchscreen monitor, the footpad, additional secondary "basic" monitors, and optionally kickpad sensors. Each "basic" touchscreen would be driven by its own Raspberry Pi 3. From a cost-sourcing point of view, the "off-the-shelf" components were going to be pricey (over $1,000 just for the monitors and Raspberry Pis). Although not extreme for prototyping, this decision rule would slow our overall commercialization process, so we decided to update to a miniature computer solution with USB peripherals.

External Data Storage

Intel NUC Mini PC

Debian (Linux) Operating System

USB Communications with Peripherals

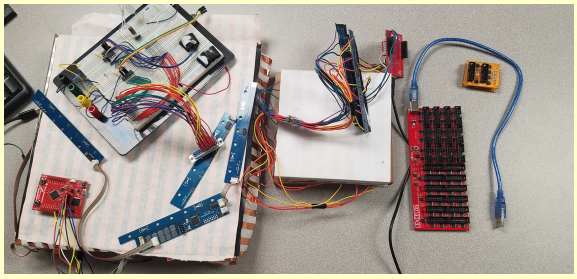
With this configuration, the primary monitor can connect to the NUC via standard HDMI. Any auxiliary peripherals, such as second touchscreens can connect via USB. The system can log significant amounts of data, which will be stored on an external USB Hard Drive such as the Seagate 2TB. USB is a ubiquitous method to connect peripherals, and the Debian OS is known for being very stable (and free). So the new design requirement is: can we buy or develop peripherals that can transmit data over USB? For secondary monitors, we need basic resolution and touchscreen capabilities, and have tested a Xenarc product successfully. The "guts" requires a DisplayLink microcontroller (DL-165) which has semi-proprietary drivers to implement the USB 2.0 protocol. As part of the project, we were able to get the above system properly configured to utilize the primary screen and the touchscreens.

We hired a group of ECEs to work on the basic platform to communicate *any peripheral with the NUC via USB*. ECEs are Electronics/Computer Engineers that have experience in both hardware with its firmware, and software (similar to Computer Scientists). As part of this engineering-rampup process, we brainstormed and came up with several potential peripherals to use with our system. In addition to the floor mat, we devise a similar hand mat, a fine-motor station, ideas for grip control (squeeze and measure air pressure), and an inexpensive punch/kick pad using a piezo sensor (inexpensive, less than 25 cents each).

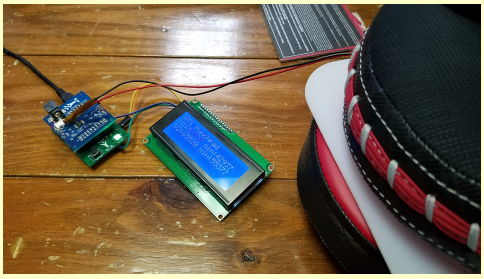
USB is an abbreviation for Universal Serial Bus and was designed in the 1990s specially to allow for the standardization of interfacing. As we learned, a limitation of the USB protocol is the identification of *multiple devices of the same type*. USB identification consists of a Vendor ID and a Product ID. Out system may have 20 or more of the same peripheral that needs to interface correctly with the NUC. As such, we developed as part of our hardware platform a method to uniquely identify a given hardware peripheral with a globally unique identifier (GUID). Our internal servers will hand out unique IDs when the hardware is initially created, and with this GUID spatial positioning can be reconciled with the NUC (e.g., identifying which punch pad is in position X within the system).

At this point, we experimented with various microcontrollers and IDE platforms to develop our prototypes. We decided to use the TIVA TMC4 and Atmel SAMD11/SAMD21 as our original microcontrollers. For the floor/handmat, we have developed two revisions using the TIVA, one using the SAMD11, and one using the SAMD21 (our latest version). As we iterated through our options, we decided to pick a single microcontroller platform (the SAMD11/SAMD21) to minimize the firmware requirements. The SAMD11 has 4K of RAM, 16K of flashable ROM, a 32-Bit ARM Cortex M0+ processor. The 24-pin version we use can be purchased on Digi-Key for less than $1.50 per microcontroller.

As we further explored *how* to spatially identify peripherals, we decided we need at least a button and a light as part of a service panel to develop a “peripheral-identification” protocol. In addition, we have included a rewritable EEPROM to store the GUID aforementioned. The design requirements have led to the development of our own “Arduino-esque” hardware platform we call IDK, as in ”I do know.” We have developed two IDK boards, one for the SAMD11 and one for the SAMD21. Similar to other DEV boards, there are pin headers for custom applications. We are also developing daughter boards for our peripheral applications with prototypes for the punchpad and handmat having significant advancement. For example, the punch pad electronics. The IDK-SAMD11 is 40mm wide by 60mm long. The punchpad daughter board is 40mm wide by 50mm long. Each board has strategically placed 2mm holes in the 4 overlapping corners for assembly. We are in the process of designing a plastic enclosure to house this implementation. With standard 2.54mm header pins, the boards make a strong male-female connection and will be about 20mm tall with a clear plastic encasement.



**Handmat iterations**



**Punchpad iteration (v003), current is v004.**

The next key decision was related to the hardware. Since the SAMD11 has a very small flashable ROM, we decided to pass on Arduino, Atmel's ASF, and Microchips IDE (Microchip recently acquired Atmel). We experiemented with each of these options, but soon learned the 16KB flash size was extremely limiting, especially for what we currently have on the IDK:

**[Microcontroller:]** **SAMD11.** 32-Bit ARM Cortex M0+ Processor, 4K RAM, 16K ROM, 24-pins

**[USB:]** USB powers the microcontroller. SAMD11 has preconfigured pins designed for one USB FS Device (DP/DM/SOF) [datasheet].

**[EEPROM:]** Flashable storage for the GUID, currently 256K using i2c protocol. The EEPROM is partitioned into 4 blocks: GUID (4KB), System (32KB), Application (Remaining). The SAMD11/21 may have a standalone 128-bit onboard solution as well.

**[Flashing/Debugging:]** UART and JTAG allow the microcontroller to be programmed. Optional components so they can be temporarily attached for flashing/debugging/troubleshooting or permanently attached for a DEV board.

**[Standalone:]** A second element on the i2c bus allows for a LCD screen to be programmed to output data in "standalone" mode (as opposed to standard data-streaming via USB to the NUC logging on the external drive).

**[Button/Light:]** Service panel to perform calibration on positional identification (spacially know which punch pad is where).

**[Light Passthrough:]** A digital protocol to drive multiple LED lights with a single digital pin. We are using SK6812 and the service panel is the first LED in an array. The punchpad has an additional 40 lights that can be controlled through the same protocol. These LEDs are RGBw programmable to customize the color. With the appropriate TCC timer, animations can be easily implemented with minimal code. This protocol has been developed and stored in a DMAC to prevent 'bit banging' and to enable asynchronous, nonblocking data streams.

**[Button/Light:]** Service panel to perform calibration on positional identification (spacially know which punch pad is where).

**[Sound:]** Also optional, play simple MIDI-type sounds on the IDK with necessary firmware.

For the IDK-SAMD11, our current compressed code base (level 2) is about 12KB. These compact code base with IDK platform represents the fundamental DNA of our sytems architecture which we have named "XELA." Similar to "Cortana, Alexa, and Siri", XELA will ultimately be our computer AI to interface with the system users. However, its DNA consists of efficient code that is easy to support and maintain. The first layer of this efficiency is the firmware-level code. We contacted a developer at Atmel about a smaller code base than ASF, and he directed us to a project he had developed to meet our basic requirements (collect sensor data and send over USB). From that starting point, we have written our code base to get data from sensors and stream via USB our output on the LCD panel.

In summary, our punchpad components now cost us about $10 each in materials. We can have 20 punchpads in the system for what two "Raspberry Pi with Touchscreens" was costing in our earlier design. This solution will be a device that we can sell at a more reasonable price.

We have developed basic drivers for our IDK platform that work on most OS platforms: Debian (Linux), Microsoft, and MacOS. The drivers will identify the device (e.g. as a punchpad) in space (top-left or position 18) and start/stop a service to stream and log the data to the NUC. A simple web application using server-side events interface with this data to enable interaction–the basis of our PD-specific platform. We have begun initial gamefication development to make the sessions more iterative and enjoyable.

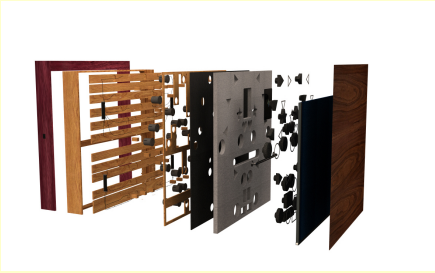
During Phase I, we had an initial cabinet built to house the electronics. This first build led us to review industrial design principles related to our primary offering, an in-home virtual PT coach. In addition, to allow for more rapid prototyping, we are preparing to prepare our PCBs inhouse. A single system will have 20 or more punchpads, and at that point, making the PCB more efficient requires some efforts. We have built a reflow oven to 'pick and place' the components on our custom-designed boards.

**Cabinet v001 iteration**

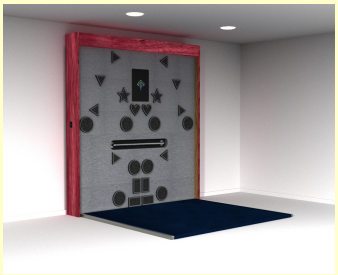
**Reflow Oven**

## 3.2 System Design Concept

The system will take about 7 feet by 7 feet by 7 feet when opened. Initially, we considered using a armoire with doors that opened, about 40 inches wide when closed and 16 inches deep. For various reasons of design, we replaced this approach with a wider, more lower profile wall-attached footprint. Now it is the full 7 feet wide and about 8 inches deep. When opened, the floor mat lowers into place as seen in the animation below.



**Exploded View Details**



**System**

**Complex Task Performed**

The system will be installed by our technicians into a user's home permanently attaching to a wall in an area chosen by the user. The system will require one power outlet and all of the electronics will be housed with the system. The tasks performed will generally be visual cues as seen in the example, the screen displays two yellow stars and the word grass (which correlates with the lower area, green square punch pad). Speakers have been built into the design to allow audio-only cues or multimedia cues combining audio with visuals.