Midas: Fabricating Custom Capacitive Touch Sensors to Prototype Interactive Objects

Valkyrie Savage
Xiaohan Zhang
Björn Hartmann

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-43
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-43.html

April 18, 2012

Midas: Fabricating Custom Capacitive Touch Sensors to Prototype Interactive Objects

Valkyrie Savage, Xiaohan Zhang, Björn Hartmann
Computer Science Division & Berkeley Institute of Design
University of California, Berkeley
valkyrie@eecs.berkeley.edu, zhangxiaohan@berkeley.edu, bjoern@eecs.berkeley.edu

ABSTRACT
An increasing number of consumer products include user interfaces that rely on touch input. While digital fabrication techniques such as 3D printing make it easier to prototype the shape of custom devices, adding interactivity to such prototypes remains a challenge for most designers. We introduce Midas, a software and hardware toolkit to support the design, fabrication, and programming of flexible capacitive touch sensors for interactive objects. With Midas, designers first define the desired shape, layout, and type of touch sensitive areas in a sensor editor interface. From this high-level specification, Midas automatically generates layout files with appropriate sensor pads and routed connections. These files are then used to fabricate sensors using digital fabrication processes, e.g. vinyl cutters and circuit board printers. Using step-by-step assembly instructions generated by Midas, designers connect these sensors to our microcontroller setup, which detects touch events. Once the prototype is assembled, designers can define interactivity for their sensors: Midas supports both record-and-replay actions for controlling existing local applications and WebSocket-based event output for controlling novel or remote applications. In a first-use study with three participants, users successfully prototyped media players. We also demonstrate how Midas can be used to create a number of touch-sensitive interfaces.

ACM Classification: H.5.2 [User Interfaces (D.2.2, H.1.2, I.3.6)]: Miscellaneous.

General terms: Design, Human Factors

Keywords: Fabrication, Prototyping, Design Tools, Capacitive Touch.

INTRODUCTION
Ubiquitous, cheap microprocessors have led to a vast increase in consumer products with built-in digital user interfaces. Many of these devices — thermostats, kitchen appliances, game controllers, and personal medical devices, to name a few — rely on touch sensing to provide input to their user interfaces.

Figure 1: Midas enables users to define discrete and continuous touch sensors with custom shapes and layout. It generates fabrication files and assembly instructions. Designers can also define the interaction events of their prototype.

Digital fabrication processes such as 3D printing and CNC machining are making it easier to prototype the form of such products, enabling designers to go directly from a 3D model in software to a physical form. In addition, user interface prototyping tools have lowered the threshold to connect new sensors to graphical user interfaces. However, one main limitation of current toolkits such as Phidgets [8], d.tools [11], Calder [17] or Arduino [2] is that they rely on pre-packaged, off-the-shelf sensors, such as momentary switches or slide potentiometers. Using such pre-packaged sensors has important drawbacks. They *constrain exploration*: pre-defined physical form factor restricts the space of realizable designs, for example buttons may be too bulky or too small, or sliders may be too long or too short. They lack *physical flexibility*: rigid sensors cannot be easily applied to non-planar surfaces or added to existing objects. A large *gulf of execution* remains between digital design files and the physical prototype: physical sensors have to be manually placed, routed, and wired. This process is tedious and error-prone; it is easy for digital design files and physical prototypes to differ. While recent research has introduced systems to create touch sensors [12, 13, 30], their efforts have focused on rapidly

constructible, low-fidelity prototypes. In contrast, our work leverages digital design tools and enables designers to use the growing range of fabrication processes to create customized, durable touch sensors.

We take inspiration from the success of Graphical User Interface editors. These editors enable designers to specify layout, size, and characteristics of widgets. They isolate designers from having to specify the "plumbing" that connects widgets to event callbacks. *Our research goal is to make the creation of physical touch-sensing interfaces as fluid as the creation of graphical user interfaces in GUI editors.*

To this end, we introduce Midas, a software and hardware toolkit for rapidly designing, fabricating, and programming flexible capacitive touch sensors (see Figure 1). With Midas, designers first define the desired shape, layout, and type of touch sensitive areas in a *sensor editor* interface. Designers can choose from discrete inputs (buttons), sliders, and two-dimensional grid sensors. For discrete inputs, designers can import graphics to define custom shapes; other types are adjustable in size and aspect ratio. Once a designer has settled on a layout, Midas automatically synthesizes appropriate capacitive touch sensor pads and routes connecting traces from sensor pads to a central touch sensing module using a circuit board grid routing algorithm [16]. Midas then generates appropriate machine instruction files and step-by-step human instructions that designers use to fabricate the sensors using rapid manufacturing techniques. Our prototype cuts sensors from adhesive-backed copper foil and insulating mask layers from vinyl on a commercial vinyl cutter. We also demonstrate how the process can be applied to circuit board milling machines. Designers then transfer their flexible, adhesive-backed sensors onto the target object; and connect the fabricated sensors to a small microcontroller using the routed connections. The microcontroller detects touch events using charge-transfer sensing [24] and forwards events to a connected PC. Once assembled, designers can define interactivity on the PC using the sensor editor: Midas supports both record-and-replay actions to control existing applications, and WebSocket event output to control novel and remote applications. WebSockets enable designers to write touch-sensitive applications using standard Web technologies (HTML and Javascript).

We demonstrate the expressivity of Midas through a number of touch-sensitive interfaces. The authors used Midas to create several touch-sensitive interfaces, including recreating prototypes of existing and published systems. In an informal first-use study, three participants successfully prototyped media player peripherals, also using Midas.

The main contributions of this paper are: 1) a novel method to create custom-shaped, flexible capacitive touch sensors based on synthesizing sensor pads and auto-routing connections from a high-level graphical specification; 2) a design tool that uses this method to enable users to to both fabricate and program custom-shaped sensors; 3) an evaluation that demonstrates Midas's expressivity and utility to designers.

The remainder of this paper is organized as follows: we first review related work, then describe how designers work with Midas. We present Midas's architecture and important limitations, describe interfaces created by the authors with Midas, report on a first-use study, and conclude with a discussion of future work.

RELATED WORK

Midas builds upon prior work in physical computing toolkits and direct touch sensing. We also discuss alternative remote touch sensing approaches for prototyping and Midas's relation to circuit board design and fabrication tools.

Physical Computing Toolkits

A substantial body of work has introduced toolkits that facilitate connecting different types of sensors and actuators to user interfaces. Some research targets software developers and enable them to extend their reach into the physical world via object-oriented wrappers to physical components [8, 19]. Other research has explicitly targeted prototyping by interaction designers [3, 10, 11, 14, 17, 18, 21, 22]; such projects employ direct manipulation interfaces or programming by demonstration [7, 10] to enable experimentation by designers who do not write code. A third set of projects are aimed at hobbyists and learners: such systems focus on helping users learn how to code, and on fostering amateur communities [5, 25].

Many prior toolkits rely on a library of prepackaged hardware sensors, which bring physical constraints with them; it is difficult to create touch sensors of custom sizes, and it may be difficult to attach sensors to existing objects. Using flexible multi-layer conductive substrates [28] or embedding radio transceivers in every component [17] gives designers more freedom of placement, but the components themselves are still fixed in size and shape. In contrast, Midas focuses on only one type of input – capacitive touch sensing – but provides explicit support for defining arbitrary sensor shapes and layouts.

Direct Touch Sensing

Most closely related to our project, BOXES [13] enables designers to create custom touch sensors from office supplies (e.g., thumb tacks); it also enables designers to program responses to sensor input using record-and-replay of GUI actions (e.g., mouse clicks). We leverage the same programming approach but target higher-fidelity prototypes: Midas supports continuous as well as discrete (step-wise) sliders, it enables construction of more durable prototypes, and permits designers to use digital design tools to create custom sensor layouts. In addition, Midas can also output events to web applications.

We also draw inspiration from projects that introduce various forms of "sensor tape" touch-sensitive material that can be unrolled and cut to size to fit custom applications. TactileTape uses resistive graphite to create custom-sized slide potentiometers [12], while Wimmer's time-domain reflectometry approach [30] measures electric pulse reflections in a wire to localize touch points. Both techniques lend themselves to low-fidelity prototyping, but do not contribute design tools to support arbitrary sensor shapes.

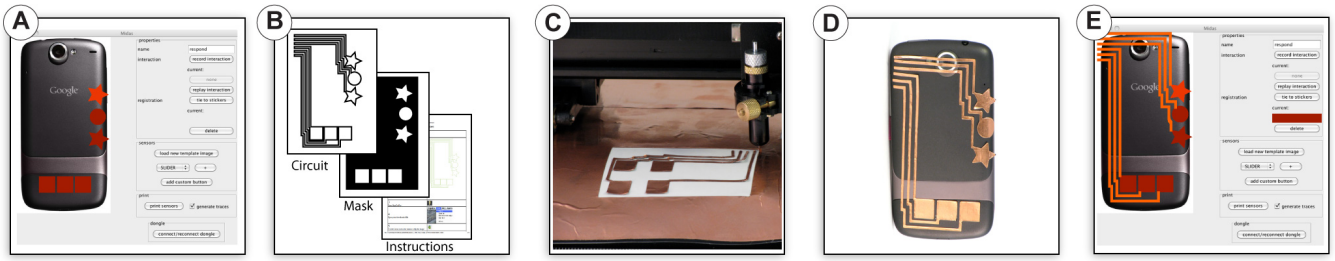


Figure 2: The Midas workflow: (A) A user creates and positions touch elements in the sensor editor. (B) Midas generates fabrication files and instructions. (C) The user fabricates the sensor on a cutting machine. (D) The user adheres sensors to his target object and connects the Midas touch controller. (E) The user authors interactivity in the sensor editor.

"Remote" Touch Sensing

A complementary approach is to sense touch *remotely*, through sensors placed in the environment, instead of the touched surface itself. Depth cameras can be used to segment fingers from background and detect touch events [29]. OmniTouch [9] uses this technique to make arbitrary surfaces touch-sensitive. Real-time motion capture systems (e.g., Vicon cameras and marker) have also been used to detect touch, e.g., in DisplayObjects [1]. Remote sensing has the advantage that surfaces and objects do not have to be specially prepared. The main disadvantages are that the technique requires an instrumented environment, or additional body-worn hardware. In addition, infrared-based systems often do not work outdoors; they may struggle with occlusion; they need significant processing power that may not be available in mobile scenarios; and their created prototypes' software has to be rewritten to change sensing technologies when when moving towards production.

Design Tools for Fabrication

The placement of sensing areas and routing of connections in Midas resembles the workflow of design tools for printed circuit board design, e.g., Eagle [6]. We share the use of auto-routing algorithms with such tools. However, circuit board design tools are usually based on a library of fixed-size components. Midas does not have these restrictions because pads are used for human input sensing, not for placing electronic components. Users can resize and easily import custom shapes in Midas. Finally, our work shares motivation with other applications that leverage digital fabrication equipment. For example, Igarashi's work on high-level design tools for plush toys [20] and chairs [26] also generate fabrication files from high-level design specifications. Our contributions are complementary to and independent from this research.

DESIGNING WITH MIDAS

This section demonstrates the interface affordances and the workflow of Midas (Figure 2) with a concrete running example: A designer would like to explore back-of-device and bezel interactions for a mobile phone. In particular, she would like to explore scrolling through a list of emails with a slider on the back of the device, and opening, replying and deleting messages through touch sensitive areas on the bezel under the phone user's thumb.

Drawing Sensors

Users start by loading a 2D projection of the physical prototype they want to augment into Midas's sensor editor. The sensor editor (Figure 3) allows a user to create the sensor layout, and define interactive behavior for each sensor. The background device image helps designers with correct scaling and positioning. Currently, Midas supports 2D projections for flat surfaces. Future work will investigate painting sensing areas onto 3D models. Sensor positioning works analogously to a GUI editor; users choose touch sensor types and drag them to the desired location on the canvas. Midas supports individual discrete buttons, one-dimensional sliders, and two-dimensional grids. Buttons can take on arbitrary shapes users can import any desired graphics file (in PNG format). Sliders and pads are currently restricted to rectangular shapes; however, their size, aspect ratio, and positioning can be modified.

In our phone example, the designer creates one slider and three discrete buttons. For the individual buttons, she loads custom shapes that she created in a drawing program.

Fabricating and Applying Flexible Sensors

Once users have a complete layout, clicking on the *print sensors* button generates fabrication files for the current layout. First, certain components are automatically split into mul-

Figure 3: Midas's sensor editor takes its cues from GUI editors: designers first lay out sensing areas in the sensor editor; they later define interactions for each sensor using a property inspector.

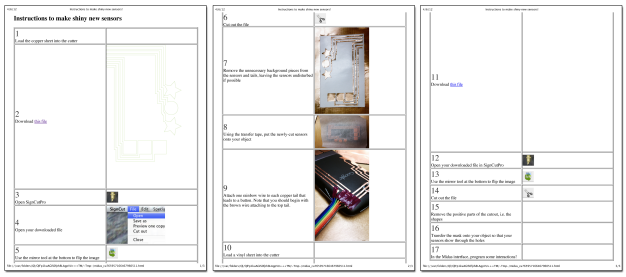


Figure 4: Auto-generated step-by-step instructions in HTML format lead the user through the fabrication and assembly process. Relevant design files are hyperlinked to specific steps; instructions also include general help on process, e.g., how to use transfer tape to apply a sensor onto an object.

tiple sensing pads. For instance, the slider in our example generates four interdigitated pads (see 6, third template) for continuous finger tracking, while 2D grids result in two different layers of triangular pads. Second, Midas generates conductive traces that will connect each of the pads to Midas's small hardware touch controller. Such connection routing determines the exact position of each touch area. Should the user want to experiment with positioning, Midas can also skip the routing step and only generate the individual touch pads. However, the user must then manually connect hookup wires to each touch pad and register them in the interface.

The pad creation and routing step generates a set of graphics files (in SVG format) and an instruction sheet (in HTML) which appears in the user's browser (see Figure 4). This sheet contains step-by-step instructions describing how to fabricate the generated files. For our implementation, instructions include which SVG files to cut in which material and how to transfer the cut designs to the prototype object.

In our phone example, the designer generates one SVG file for the touch areas, and one mask file to cover traces, which can prevent accidental touch events. Following the generated instruction web page, she first feeds a sheet of copper foil into her vinyl cutter and cuts the corresponding SVG file. She then substitutes a vinyl roll for the copper foil and cuts a mask layer. As both materials have adhesive backing, she sticks the copper and vinyl layers onto the phone she wishes to modify. Once the adhesive layers are applied, she tapes the end of the routing leads to the Midas touch controller module, which is plugged into her computer via USB.

Connecting Hardware to Software

Touch sensing is performed by a dedicated touch controller circuit board. Users do not have to program or assemble any electronics - they may treat the entire setup as a prototyping "dongle". Users do have to connect the end of the traces to the controller through a short rainbow ribbon cable, either by taping the cable leads onto copper traces, or by soldering them.

To complete a prototype, users return to the sensor editor. In many toolkits, mapping hardware components to named objects in software can be error-prone it is easy to swap wires or connect to an incorrect pin. If the user prints a fully routed

Figure 5: Users start to record GUI interactions in the sensor editor (A); they can for example activate the browser, enter text (B), and click on a search result (C); before concluding the recording (D). This sequence of actions can then be triggered by a touch event.

design, Midas already knows and generates instructions for how pins are mapped onto touch areas. If the user decided to wire the design herself, this mapping has to be authored. Midas uses *guided demonstration* to assist with authoring this mapping. For buttons, the user selects an input element in the UI and clicks the *tie to stickers* button; next she touches the corresponding copper sticker. Midas listens for status change events and assigns the correct hardware pins to that button. Midas registers sliders similarly: users are asked to simply swipe a finger along the slider.

Adding Interactivity

Designers have two options for authoring interactivity: record-and-replay of mouse and keyboard events (a strategy adopted from BOXES [14] and Exemplar [10]), or touch event output to control other applications over WebSockets. To record and replay interactions, designers select a sensor in the editor, then click on the *record interaction* button. They can then control any open application (e.g., start or stop a media player application, or drag a volume slider). Midas records the generated keyboard and mouse events and can replay them later in response to touch input (Figure 5).

The types of desktop UI actions that can be executed depend on the button type. Individual buttons can be tied to an *interaction script*, a sequence of keyboard and mouse events recorded by the user. Sliders are linked to exactly one horizontal or vertical line on the screen to be controlled by clicks along its length. 2D grids can control a 2D area on the screen analogous to a slider area. For sliders and grids, the user must capture the location on the screen that she wishes to control with the slider or grid. This is done by simply clicking at each end of the slider or in opposite corners of the grid, as Midas prompts. As the user adjusts the sensitivity (in number of discrete buttons) of the slider or pad to be printed, the interaction with the captured on-screen slider or pad becomes more fine-grained, as well

Record-and-replay does not require programming, but it is brittle; changes in application layout or response latency can break a recorded sequence. To let users author more robust interactions, Midas uses WebSockets to send touch events into other applications. This option requires programming, but Midas's use of WebSockets enables users to work with the languages many are most familiar with: HTML and JavaScript.

In our phone example, the designer chooses WebSockets as she wants to demonstrate how touch events can control an

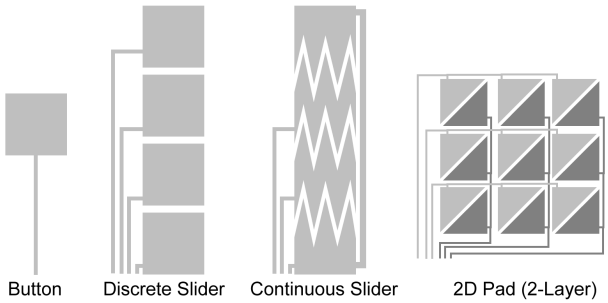


Figure 6: Midas can generate four different types of sensors: discrete buttons, discrete sliders, continuous sliders, and 2D pads. The pad uses row-column scanning and requires multi-layer construction because traces cross.

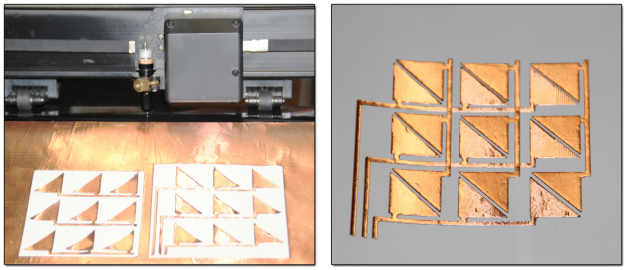


Figure 7: 2D grid sensors are fabricated in two different layers that are then superimposed. Because each copper layer has a vinyl backing, no other inter-layer masking is required.

mobile email application. She creates a mockup in HTML and writes JavaScript functions to receive touch events.

ARCHITECTURE AND IMPLEMENTATION

This section describes the architecture and algorithms underlying the Midas system.

Generating Sensor Pads

The Midas sensor editor supports placement of four types of touch sensors: discrete buttons, two types of one-dimensional sliders, and 2D grids. The resolution of pads and segmented sliders can be set through a parameter in the sensor editor. The current editor is written in Java using the Swing GUI Toolkit. Figure 6 shows example templates for each sensor type. The two types of sliders are based on different sensing approaches. The first, *segmented slider*, is made up of individual rectangular touch segments. Users can specify how many segments the slider should have. *Continuous sliders* offer finer resolution, but require a different detection approach. We use Bigelow's design of interdigitated electrodes [4]. In this design, as the finger slides across the pads, the surface area of the pads underneath the finger changes as pad digits get smaller or larger. Because capacitance is proportional to contact surface area, the capacitance of each segment changes throughout the finger's slide. While finer in resolution, only one such slider is supported by our current sensing hardware. Increasing the number of supported sliders is possible with additional engineering effort.

2D pads leverage row-column scanning to reduce the number of connection traces necessary. For example, a 5x5 array would need 25 individual traces, but only $5 + 5 = 10$ traces in row-column mode. This design requires a dual-layer construction where horizontal traces are isolated from vertical traces. We feed copper foil applied to vinyl foil into our cutter, so each layer is already on an insulating substrate. Designers thus first apply the bottom conductive layer, then place the top layer directly over it (see Figure 7).

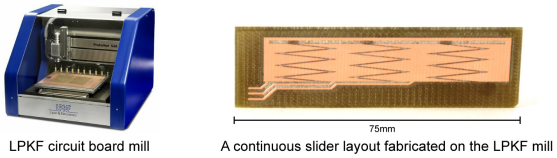
To generate a mask layer that covers the topmost copper traces, we generate a design file that contains pads from all layers, but without any traces. This layer is cut in vinyl. Whereas for other layers, designers transfer the pads and traces, for the mask layer they peel and transfer the surrounding, "background" shape (see Figure 12, left).

Routing Pads to the Touch Controller

Midas employs an auto-routing algorithm to generate conductive traces that connect electrodes to the Midas touch controller. We implement Lee's maze routing algorithm for single layer paths [16]. For 2D pads, we perform two independent routings: once for the row layer, and once for the column layer. Our current algorithm does not generate vias (connections between two different conductive layers). When autorouting fails, we employ an iterative search by adjusting the position where traces connect to sensor pads, routing the sensors in a different order, or moving the position where target traces connect to the touch controller. In our experience, this basic routing algorithm has performed adequately; it could also be replaced it with more sophisticated routing techniques in the future.

Fabrication

Midas generates vector graphics files in SVG format for the different electrode and mask layers. These files can then be used to control digital fabrication processes. Our prototype currently cuts conductive, adhesive-backed copper foil on a commercial vinyl cutter a plotter with a cutting knife instead of a pen. This medium has multiple advantages. First, it is cost-effective and readily available: vinyl cutters are in the same price range as laser printers (ours, a tabletop model with a 14-inch bed) cost $200); and copper foil costs a few dollars per foot. Second, copper has excellent conductivity. Third, flexible, adhesive foil is easy to apply to non-planar surfaces. However, there are important drawbacks as well. Most importantly, the subtractive cutting process and manual weeding (removing background material) determines a minimum feature size for traces as thin geometric features can break during transfer, and the weeding process can be te-



LPKF circuit board mill

A continuous slider layout fabricated on the LPKF mill

Figure 8: Example of a capacitive touch sensor manufactured through an alternative process of circuit board milling.

dious and time-consuming. We found the most success came from adhering the copper sheets to vinyl sheets and cutting both layers at once. This setup has the added benefit of allowing designers to prototype capacitive touch interactions on conductive surfaces (e.g., the backside of an iPhone) because the vinyl is an excellent insulator.

Alternative fabrication processes may be preferable to copper foil cutting when higher precision or durability is required. Two promising approaches are circuit board milling, which can produce smaller features but is limited to rigid boards; and conductive inkjet printing, which can produce the smallest features, but is not yet available to many end users. As a proof of concept, we produced a touch sensor on an LPKF circuit board milling machine (see Figure 8).

Capacitive Touch Sensing

The Midas touch controller (Figure 9) is based on an Atmel microcontroller board [27] and capacitive touch sensing chips from Quantum (QT1106) and Freescale (MPR121) Semiconductors. For discrete inputs, both chips rely on charge-transfer sensing using single-wire electrodes: the electrodes are part of a simple RC circuit in which an output pin is set high, and time is measured until an input pin also reads high. This time is proportional to the capacitance of the circuit: when a person touches an electrode, the circuit capacitance and the charge transfer time both increase. The Quantum chip also implements Bigelow's design to extract continuous position readings from interdigitated electrodes [4]. The microcontroller runs software written in embedded C to interface with the sensor chips and communicates touch data to a connected computer using the USB HID protocol. It recalibrates the touch sensing chips periodically to ensure floating sensor values do not lead to erroneous touch readings.

Event Output

Once interface scripts are assigned to buttons, Midas's Arduino connection listens for events from the printed touch sensors. When a signal matching one of the saved interactions is encountered, the associated script is activated.

Record-And-Replay. In record-and-replay, the user demonstrates a touch interaction and then records a corresponding action on their desktop that should be triggered by the touch action. Early prototypes of Midas used Sikuli, a visual scripting language based on computer vision analysis of screenshots to define desktop actions [33]. While more robust to GUI changes than hardcoded click locations, Sikuli was designed for automation scripts rather than interactive control, and the latency of invoking and executing scripts was too high. Our current prototype, uses the Java Robot API [15] for literal click captures as in the BOXES system [14]. The Robot captures and replays both click locations and typing.

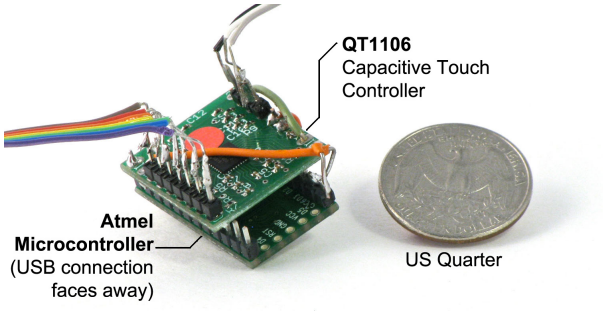


Figure 9: The Midas touch controller board uses a commercial capacitive charge transfer detection chip to sense touch events. Events are relayed to a computer via a mini USB connection on the back. The ribbon cables are used to connect to the end of routed traces. A US quarter is shown as a size reference.

WebSocket Communication with Web Applications. Record-and-replay is restricted to applications that run on the same machine as the sensor editor, and it is limited to mouse and keyboard event injection. To surmount these limitations, Midas can also export touch events via a built-in server to remote clients using the WebSockets API. For example, an application running on a smart phone can open a WebSocket connection to Midas and receive a callback whenever any Midas button, slider or grid changes. The callback function will also receive an event object that describes which sensor changed state, and the value of that new state (e.g., on/off, or a slider value).

Our WebSockets server is implemented in node.js using the socket.io library. We chose WebSockets because it offers full-duplex communication at low latencies, and, more importantly, they are supported by modern web browsers. This means that designers can author user interfaces that respond to Midas touch input in HTML and JavaScript. There are two main benefits to these technologies: (1) many designers are already familiar with them from web design; (2) developed interfaces can be deployed on any device that has a compatible browser, even if that device does not offer a way to directly connect external hardware. For example, it is difficult to directly connect sensors to mobile devices such as Apple's iPhone or iPad.

With our WebSockets architecture (Figure 10), designers open their phone's web browser, and enter the URL of an HTML file they have placed in the Midas server directory. This file opens a socket connection from the phone browser to the server. When the server receives events from the Midas touch controller, it forwards them to the client, which can then show visual feedback.

LIMITATIONS

Our current prototype has some important limitations. A few are inherent to our chosen approach, while others could be mitigated with additional engineering.

First, the current manufacturing process places certain physical constraints on realizable designs. Both the accuracy of the vinyl cutter on copper and challenges in weeding and transferring cut designs currently restrict traces to approximately 2mm minimum thickness. A higher-quality cutter could reduce this threshold. Our cutter also has difficulties cutting acute angles such as those in the interdigitated slider.

Second, the touch sensing chips have limited capacity. The QT1106 has 7 discrete sensing pins and additional pins for one continuous slider; the MPR121 has 13 discrete inputs.

The sensor editor keeps track of resources required by the current design and notifies designers if they have exceeded the capacity of the touch controller dongle. While we currently do not support using multiple touch controllers or multiple touch chips on a single controller, a future circuit board revision could offer such support. In addition, continuous sliders use different hardware resources than other inputs and therefore need to be treated differently by the designer.

Third, our touch controller must be tethered to a computer. This reduces mobility: prototypes cannot currently be tested outside the lab. Direct connections to mobile devices or integrated wireless radios could address this constraint.

Finally, Midas only supports printing of flat designs. While the resulting sensors are flexible and can be applied to non-planar surfaces, the sensor editor does not yet support designers in correctly laying out shapes and traces on such objects.

EVALUATION

To understand the user experience of working with Midas, and to assess its expressivity, we implemented some touch-sensitive applications and conducted an informal first-use study of Midas with three participants.

Example Applications

To demonstrate the expressivity of Midas as a prototyping tool, we re-implemented several touch-based interactive systems that would test the full extent of Midas's components: WebSockets, record-and-replay, and a range of sensor types.

Text Entry. Wobrrock's EdgeWrite [32] is a unistroke text entry technique based on activating a series of corner points of a rectangle. Wobbrock demonstrated that this technique can be implemented using four discrete capacitive touch sensors [31]. We printed four discrete buttons and a mask with Midas, attached them to the back of a smartphone, and implemented the EdgeWrite recognition algorithm in Javascript (Figure 11). Using socket events, we demonstrated how EdgeWrite can be used to enter text on the back of a mobile device, leaving the screen unobstructed. The implementation is functional, though latency for detecting single button presses was higher than expected (>100ms). We plan to investigate ways to increase responsiveness for buttons in future work.

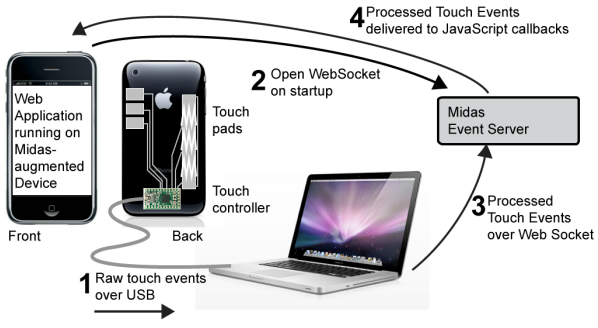


Figure 10: Midas's socket event output enables designers with programming knowledge to create web applications in HTML and Javascript that react to touch input outside the screen area of a phone or tablet.

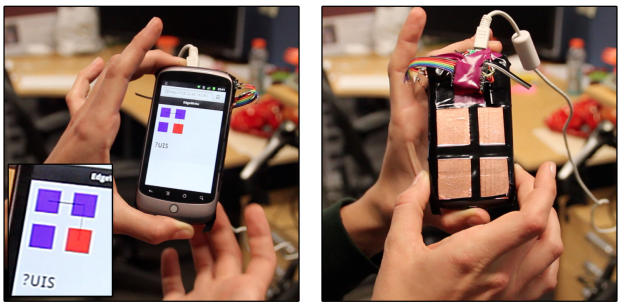


Figure 11: We implemented Wobbrock's Edgewrite on the back of a cell phone using a 2-by-2 grid and WebSocket events sent to a web page.

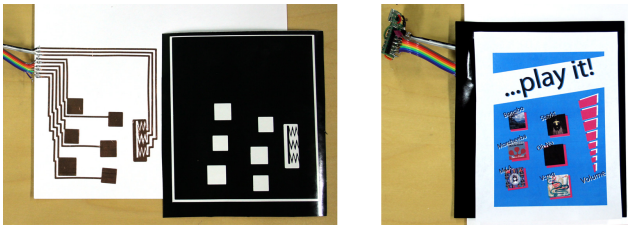


Figure 12: Our music postcard lets users sample tracks by different artists. Left: Circuit and mask layer; Right: assembled postcard.

Game Controller. To test the responsiveness of Midas's continuous slider sensing, we created a simple game controller for the video game Breakout, in which players horizontally position a paddle to bounce a ball into layers of blocks. In less than fifteen minutes, we attached the slider that we fabricated on the circuit board milling machine to a non-touch-sensitive 7-inch monitor and mapped slider position to paddle position using Midas's record-and-replay system. The slider is more responsive than individual buttons, and we were able to control the paddle accurately enough for gameplay. The slider's response is non-linear across certain regions, however, and accounting for this is left to future work.

Music Postcard. At a recent media festival, a promotional poster printed with conductive ink enabled passersby to select and play music from a number of artists by touching corresponding areas on the poster [23]. We implemented a postcard-sized version of this poster (Figure 12). We scaled back our implementation of the poster to conserve resources; large designs are possible and only restricted by the cutter's bed width. Our version uses six discrete buttons and one continuous slider to control playback and volume of music clips on a desktop computer. We again cut a vinyl mask layer to prevent stray touch events. We used Midas's record-and-replay function to remote control the iTunes music player.

Informal Evaluation

To gauge the usability of Midas, we recruited three participants to prototype a media-controlling computer peripheral. Two participants were graduate students at our university (in Computer Science and Mechanical Engineering), one was a software engineer at a local technology company. All had some prior experience with prototyping and electronics.

Procedure. Participants first received a walkthrough of the Midas UI including a simple record-and-replay task to launch a browser based on a single button input. Participants were then asked to design a physical control interface for a media player (iTunes). No other constraints were given as we wanted to encourage exploration. Each session lasted up to 60 minutes. Participants completed a post-task questionnaire with open-ended questions about interface usability and utility of the workflow.

Results. All participants successfully completed the task and designs of media players (Figure 13). Participants commented positively on how Midas aided them with the task of physical construction — both by routing connections, as well as through the auto-generated instructions. Record-and-replay was easy to comprehend and effective for the given task. Even though the task did not require programming, two participants expressed interested in receiving touch events in their own applications. We take this as corroboration for the utility of our WebSocket server.

Participants identified several areas for improvement. Two participants felt that the instructions, while helpful, did not provide sufficiently detailed information for certain steps (e.g., removing the extra material around the printed pads). Two participants requested videos in addition to static images; creating such videos would be straightforward and can likely address this challenge.

Midas's routing algorithm could not find a successful solution for one version of one participant's design. This participant was unable to identify the underlying problem that caused the failure since he was unfamiliar with auto-routing and the interface did not provide any additional information about the algorithm. In future versions of the interface we plan to include instructions how a user might adapt their design for a successful routing.

Finally, all participants requested that we provide richer feedback in the interface to convey information about touch events the hardware controller was transmitting. The identified challenges are not fundamental to our architecture and can be addressed in a future design iteration.

CONCLUSION AND FUTURE WORK

This paper presented Midas, a software and hardware toolkit to support the design, fabrication, and programming of capacitive touch sensors. Midas leverages the familiar paradigm of the GUI editor to define shape layout and interactivity of capacitive touch sensors. To help designers bridge the gap between digital designs and their physical realization, Midas auto-routes connections, generates instruction sheets and offers sensor registration by demonstration. Our informal evaluation suggests that working interactive applications can be built with our current implementation and that the interface is accessible for prototyping.

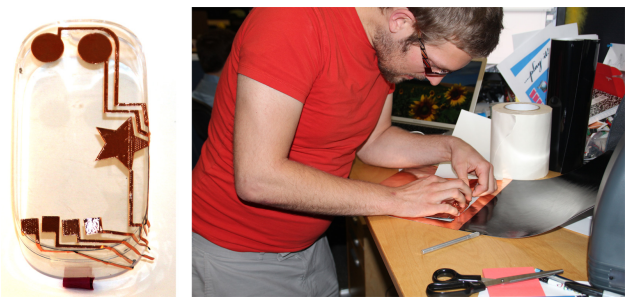


Figure 13: A study participant's sensor layout for a PC peripheral.

Our plans for future work seek to address some of the identified limitations, and also expand into new application areas. First, we would like to explore methods to *paint* touch-sensitive areas directly onto 3D CAD models and then correctly synthesize pads and routes for such 3D objects. Such a function could be delivered as a plugin for CAD software such as SolidWorks or AutoCAD. In addition, we are actively investigating different fabrication processes and plan to continue our exploration of conductive ink printing. Finally, we would like to expand our scope beyond capacitive charge-transfer sensing. For the longer term, we are interested in more closely integrating user interface design tools with digital fabrication tools.

References

1. Eric Akaoka, Tim Ginn, and Roel Vertegaal. Display-objects: prototyping functional physical interfaces on 3d styrofoam, paper or cardboard models. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, TEI '10, pages 49–56, New York, NY, USA, 2010. ACM.
2. Arduino physical computing platform. http://arduino.cc. Accessed: July, 2011.
3. Daniel Avrahami and Scott E. Hudson. Forming interactivity: a tool for rapid prototyping of physical interactive products. In *Proceedings of the 4th conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '02, pages 141–146, New York, NY, USA, 2002. ACM.
4. John E. Bigelow. Capacitive touch control and display - us patent 4264903, 1981.
5. Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. The lilypad arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 423–432, New York, NY, USA, 2008. ACM.
6. CadSoft EAGLE PCB Software. http://www.cadsoftusa.com/. Accessed: April 2012.
7. Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. a cappella: programming by demonstration of context-aware applications. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '04, pages 33–40, New York, NY, USA, 2004. ACM.
8. Saul Greenberg and Chester Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 209–218, New York, NY, USA, 2001. ACM.

9. Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. Omnitouch: wearable multitouch interaction everywhere. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 441–450, New York, NY, USA, 2011. ACM.
10. Björn Hartmann, Leith Abdulla, Manas Mittal, and Scott R. Klemmer. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '07, pages 145–154, New York, NY, USA, 2007. ACM.
11. Björn Hartmann, Scott R. Klemmer, Michael Bernstein, Leith Abdulla, Brandon Burr, Avi Robinson-Mosher, and Jennifer Gee. Reflective physical prototyping through integrated design, test, and analysis. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 299–308, New York, NY, USA, 2006. ACM.
12. David Holman and Roel Vertegaal. Tactiletape: low-cost touch sensing on curved surfaces. In *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology*, UIST '11 Adjunct, pages 17–18, New York, NY, USA, 2011. ACM.
13. Scott E. Hudson and Jennifer Mankoff. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 289–298, New York, NY, USA, 2006. ACM.
14. Scott E. Hudson and Jennifer Mankoff. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pages 289–298, New York, NY, USA, 2006. ACM.
15. Java AWT Robots API. http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html. Accessed: April, 2012.
16. C. Y. Lee. An algorithm for path connections and its applications. *Electronic Computers, IRE Transactions on*, EC-10(3):346 –365, Sept. 1961.
17. Johnny C. Lee, Daniel Avrahami, Scott E. Hudson, Jodi Forlizzi, Paul H. Dietz, and Darren Leigh. The calder toolkit: wired and wireless components for rapidly prototyping interactive devices. In *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, DIS '04, pages 167–175, New York, NY, USA, 2004. ACM.
18. Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. Dart: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pages 197–206, New York, NY, USA, 2004. ACM.
19. Nicolai Marquardt and Saul Greenberg. Distributed physical interfaces with shared phidgets. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 13–20, New York, NY, USA, 2007. ACM.
20. Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. *ACM Trans. Graph.*, 26(3), July 2007.
21. Tek-Jin Nam. Sketch-based rapid prototyping platform for hardware-software integrated interactive products. In *CHI '05 extended abstracts on Human factors in computing systems*, CHI EA '05, pages 1689–1692, New York, NY, USA, 2005. ACM.
22. Tek-Jin Nam and Woohun Lee. Integrating hardware and software: augmented reality based prototyping method for digital products. In *CHI '03 extended abstracts on Human factors in computing systems*, CHI EA '03, pages 956–957, New York, NY, USA, 2003. ACM.
23. BBC News. Playing pop music via paper posters with conductive ink, march 12. http://www.bbc.com/news/technology-17339512, 2012. Accessed: April, 2012.
24. Hal Philipp. Charge transfer sensing. *Sensor Review*, 19(2):96–105, 1999.
25. Eric Rosenbaum, Evelyn Eastmond, and David Mellis. Empowering programmability for tangibles. In *Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, TEI '10, pages 357–360, New York, NY, USA, 2010. ACM.
26. Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. Sketchchair: an all-in-one chair design system for end users. In *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, TEI '11, pages 73–80, New York, NY, USA, 2011. ACM.
27. Teensy USB Development Board. http://www.pjrc.com/teensy/. Accessed: April 2012.
28. Nicolas Villar and Hans Gellersen. A malleable control structure for softwired user interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pages 49–56, New York, NY, USA, 2007. ACM.
29. Andrew D. Wilson. Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pages 69–72, New York, NY, USA, 2010. ACM.
30. Raphael Wimmer and Patrick Baudisch. Modular and deformable touch-sensitive surfaces based on time domain reflectometry. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 517–526, New York, NY, USA, 2011. ACM.
31. Jacob O. Wobbrock. Capacitive EdgeWrite implementation. (http://depts.washington.edu/ewrite/capacitive.html). Accessed: April, 2012.
32. Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. Edgewrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, pages 61–70, New York, NY, USA, 2003. ACM.
33. Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: using gui screenshots for search and automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pages 183–192, New York, NY, USA, 2009. ACM.