# 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 many 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, as well as routing obstacles, in a sensor editor. 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 conductive ink printers. Using step-by-step assembly instructions generated by Midas, designers connect these sensors to the Midas microcontroller, 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)]: Prototyping.

**Keywords:** Design Tools; Prototyping; Fabrication; Capacitive Touch Sensing.

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

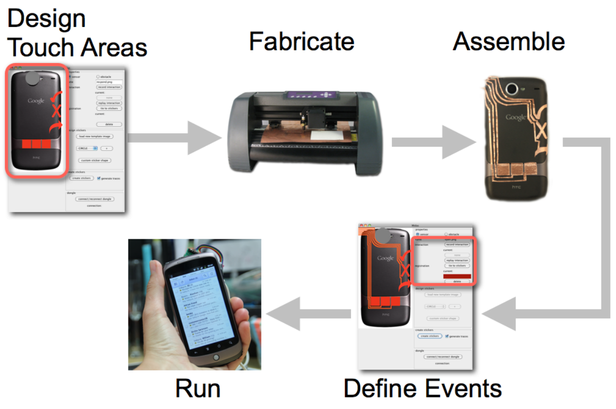


**Figure 1**: Midas enables users to define discrete and continuous touch sensors with custom shapes and layout. It generates fabrication files and assembly instructions. Designers can also define the interaction events of their prototype.

Digital fabrication processes like 3D printing and CNC machining make it easier to prototype the form of such products, enabling designers to go directly from a digital 3D model to a physical object. In addition, user interface prototyping tools have lowered the threshold to connect sensors to graphical user interfaces. However, one main limitation of current toolkits such as Phidgets [8], d.tools [11], .NET Gadgeteer [29] or Arduino [2] is their reliance on pre-packaged, off-the-shelf sensors such as momentary switches or slide potentiometers. Using pre-packaged sensors has important drawbacks. It *constrains exploration*: pre-defined physical form factors restrict the space of realizable designs. For example, available buttons can be too bulky or too small, or sliders may be too long or too short. Most sensors also lack *physical flexibility*: they are not easily applied to non-planar surfaces or added to existing objects. Finally, a large *gulf of execution* remains between digital design files and physical prototypes: sensors must be manually placed and wired one-by-one. This process is tedious and error-prone; physical prototypes can easily deviate from digital design files if a wire is incorrectly placed or forgotten. While recent research has introduced tools to create touch sensors of different shapes [12, 13, 31], their efforts focus 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 custom, durable, replicable touch sensors.

We take inspiration from the success of GUI editors. These editors enable designers to specify layout, size, and characteristics of widgets. They also isolate designers from specifying 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.*

As a first step in this direction, we introduce Midas, a software and hardware toolkit to support the design, fabrication, and programming of custom capacitive touch sensors (see Figure 1). With Midas, designers first define the desired shape, layout, and type of touch sensitive areas and obstacles in a *sensor editor* interface. Designers can choose from buttons, 1D sliders, and 2D pad sensors. For discrete (button) inputs, designers can use polygon shapes or import graphics to define custom shapes; other types are adjustable in size and aspect ratio. Once a designer settles on a layout, Midas automatically synthesizes appropriate capacitive touch sensor pads and routes connecting traces, avoiding user-defined obstacles, to a central touch sensing module via a circuit board grid routing algorithm [15]. Midas then generates layout files and step-by-step instructions that designers use to fabricate the sensors using rapid manufacturing techniques. Our prototype cuts sensors from adhesive-backed copper foil and vinyl on a commercial vinyl cutter. We also demonstrate how to use a circuit board milling machine to fabricate Midas sensors. 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 [23] and forwards events to a PC. Once assembled, designers can define interactivity on the PC using the sensor editor. Midas supports both record-and-replay actions to control existing local applications, and WebSocket event output for novel and remote applications. WebSockets enable designers to write touch-sensitive applications using standard Web technologies (HTML and JavaScript).

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

The main contributions of this paper are: 1) a novel method to create custom-shaped, flexible capacitive touch sensors by synthesizing sensor pads and auto-routing connections, as well as instructions for assembly and use, from a high-level graphical specification; 2) a design tool using this method to enable users to to fabricate, program, and share touch-sensitive prototypes; 3) an evaluation demonstrating Midas's expressivity and utility to designers.

The remainder of this paper is organized as follows: we review related work, then describe how designers work with Midas. We present Midas's architecture and limitations, describe author-created Midas interfaces, 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, enabling them to extend their reach into the physical world via object-oriented wrappers to physical components [8, 18, 29]. Other research explicitly targets prototyping by interaction designers [3, 10, 11, 13, 16, 17, 20, 21]; such projects employ direct manipulation interfaces or programming by demonstration [7, 10] to enable experimentation by designers who do not write code; our work falls in this group. 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 libraries of prepackaged hardware sensors which bring physical constraints with them; it is difficult to create sensors of custom sizes, and it may be difficult to attach them to existing objects. Flexible multi-layer conductive substrates [28] or embedded radio transceivers in each component [16] give 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 work, BOXES [13] enables designers to create custom touch sensors from office supplies (e.g., thumb tacks); it also enables them 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, enables construction of more durable prototypes, and permits designers to use digital design tools to create custom sensor layouts. Since layouts are created in software rather than cardboard and thumbtacks, Midas designs can be shared and reproduced. In addition, Midas can also output events to web applications rather than only local 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. Tactile-Tape uses resistive graphite to create custom-sized slide potentiometers [12], while Wimmer's time-domain reflectometry approach [31] 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.

### "Remote" Touch Sensing

Touch can also be sensed *remotely*, via sensors placed in the environment, instead of the touched surface itself. Depth cameras can be used to detect touch events by segmenting fingers from background areas [30]. OmniTouch [9] uses this

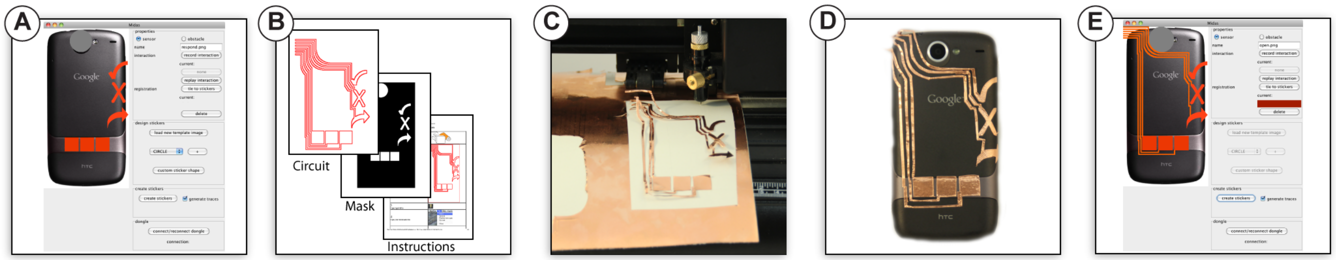


**Figure 2**: The Midas workflow: (A) A user creates and positions touch elements in the sensor editor. (B) Midas generates fabrication files and instructions. (C) The user fabricates the sensor on a cutting machine. (D) The user adheres sensors to a target object and connects the Midas touch controller.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

technique to make arbitrary surfaces touch-sensitive. Real-time motion capture systems have also been used to detect touch, e.g., Vicon cameras in DisplayObjects [1]. Remote sensing has the advantage that surfaces and objects do not have to be specially prepared. The main disadvantage is that the technique requires an instrumented environment or body-worn hardware. In addition, infrared-based systems often do not work outdoors, may struggle with occlusion, or need significant processing power not available in mobile scenarios. Prototypes that rely on remote sensing may have to be rewritten to change sensing technologies for production.

## Design Tools for Fabrication

The placement of sensors and routing of connections in Midas takes inspiration from electronic design automation tools, e.g., Eagle [6] for printed circuit board design. 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 components. Users can resize and easily import custom shapes in Midas. Finally, our work shares motivation with other applications leveraging digital fabrication equipment. For example, design tools for plush toys [19] and chairs [26] also generate fabrication files from high-level 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 scroll through a list of emails with a slider on the back of the device, and open, reply to, and delete messages via sensors on the bezel under the phone user's thumb.

### Drawing Sensors

Users start by loading an image 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 images, including flattened 3D models. Future work will investigate direct support of 3D models. Sensor positioning works analogously to a GUI editor; users choose sensor types and drag them to the desired location on the canvas. Midas supports individual discrete buttons, one-dimensional sliders, and two-dimensional pads. Buttons can take on arbitrary shapes—users can import any graphics file (in PNG format) or draw custom polygons. Sliders and pads are currently restricted to rectangular shapes; however, their size and aspect ratio can be modified to fit the requirements of the prototype at hand. Users may also define *obstacles* using the same drawing tools to restrict routing — Midas will route connections around these obstacles.

In our phone example, the designer creates one slider and three discrete buttons. For the buttons, she loads custom shapes created in a drawing program. She defines a circular obstacle around the phone's camera so the camera will not be obscured during connection routing.

### Fabricating and Applying Flexible Sensors

Once users complete a layout, clicking the *create stickers* button generates fabrication files. First, certain components are automatically split into multiple sensing pads. For instance, a slider can generate four interdigitated pads (Figure 6, third template) for continuous finger tracking, while 2D pads result in two separate layers of triangular pads (Figure 7). Second, Midas generates conductive traces that will connect each of the pads to Midas's touch controller. An ad-

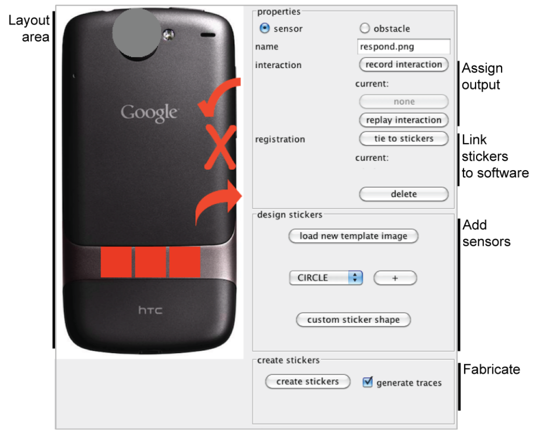
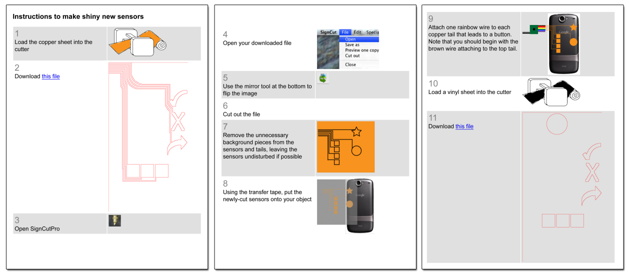


**Figure 3**: Midas's sensor editor takes its cues from GUI editors: designers first lay out sensing areas through direct manipulation; they later define interactions for each sensor using a property inspector.


**Instructions to make shiny new sensors**

1
Load the copper sheet into the cutter

2
Download this file

3
Open SignCutPro

4
Open your downloaded file

5
Use the mirror tool at the bottom to flip the image

6
Cut out the file

7
Remove the unnecessary background pieces from the sensors and tails, leaving the sensors undisturbed if possible

8
Using the transfer tape, put the newly-cut sensors onto your object

9
Attach one rainbow wire to each copper tail that leads to a button. Note that you should begin with the brown wire attaching to the top tail.

10
Load a vinyl sheet into the cutter

11
Download this file

**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 processes, e.g., how to use transfer tape to apply a sensor onto an object.

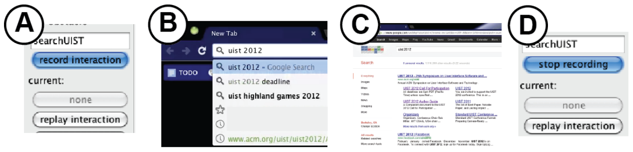


**Figure 5**: Users start to record GUI interactions in the sensor editor (A); they can for example activate the browser, enter text (B), and click on a search result (C), before concluding the recording (D). This sequence of actions can then be triggered by a touch event.

ditional mask file, to be fabricated in vinyl, includes cutouts of only the sensor shapes: it will cover the traces both for aesthetic reasons and to prevent stray touch events. Midas's connection routing determines the exact position of each touch area. Should the user want to experiment with positioning, Midas can also skip routing and only generate individual touch pads. However, the user must then manually connect wires to each sensor and register the sensor 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 to mask the traces, which prevents stray touch events. Following the generated instruction web page, she feeds copper foil into her vinyl cutter and cuts the corresponding SVG file. She then substitutes a vinyl roll 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 routed traces to the Midas hardware, which is plugged into her computer via USB. Since the design files for her prototype are digital, she also sends them to colleagues in another office for a second, remote test. With the design files and a vinyl cutter, her colleagues can then recreate a working Midas prototype.

### Connecting Hardware to Software

Midas senses touch events with 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's 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 design, Midas generates instructions for aligning touch areas with specific ribbon cable colors. If the user decided to wire the design herself, this mapping has to be authored. Midas uses *guided demonstration* to assist with this process. For buttons, the user selects an input element in the UI and clicks the *tie to stickers* button; next she touches the corresponding copper sensor. Midas listens for status change events and automatically assigns hardware pins. Midas registers sliders similarly: users are asked to swipe a finger along the slider.

Midas's editor interface displays incoming touch data visually, re-coloring touched sensors in pink on the sensor editor, to aid the user in debugging. If something goes wrong during the connection stage, it is apparent to the user. Midas also reads the data stream for common errors. If it detects that two wires may be too close together and triggering each other, or that there may be a faulty connection from a wire to the board, that information is displayed to the user in a connection status area.

### Adding Interactivity

Designers have two options for authoring interactivity: record-and-replay of mouse and keyboard events (a strategy adopted from BOXES [13] and Exemplar [10]), or touch event output to control applications via 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 pads can control a 2D area on the screen analogous to a slider area. For sliders and pads, the user must capture the location on the screen that she wishes to control with the slider or pad. This is done by clicking at each end of the slider or in opposite corners of the pad, guided by Midas prompts. As the user adjusts the sensitivity (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, also.

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 over the Internet. This requires programming, but WebSock-

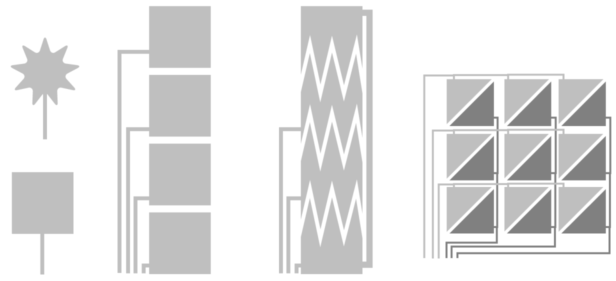


**Figure 6**: Midas can generate four different types of sensors: discrete buttons, discrete sliders, continuous sliders, and 2D pads. The pad uses row-column scanning and requires multi-layer construction because traces cross.

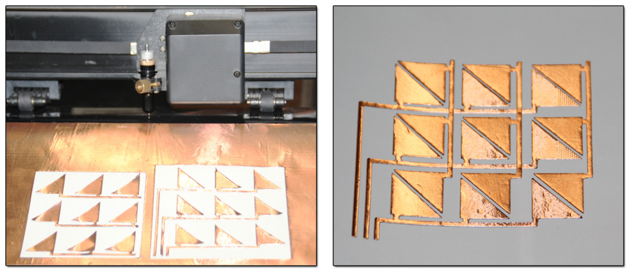


**Figure 7**: 2D pad sensors are fabricated in two different layers that are then superimposed. Because each copper layer has a vinyl backing, no other inter-layer masking is required.

ets enable designers to work in 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 a 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 four types of touch sensors: discrete buttons, two types of 1D sliders, and 2D pads. 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 specify how many segments the slider has. *Continuous sliders* offer finer resolution, but require a different detection approach. We use Bigelow's design of interdigitated electrodes [4]. In this design, as a 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 measured capacitance of each segment changes during the finger's slide. Though 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 use row-column scanning to reduce connecting traces. For example, a $5 \times 5$ array requires 25 individual traces, but only $5 + 5 = 10$ row-column traces. This design requires a dual-layer construction where horizontal traces are isolated from vertical traces. We use copper foil applied to vinyl foil in our cutter, so each layer already has an insulating substrate. Designers thus first apply the bottom conductive layer, then place the top layer directly over it (see Figure 7).

To create a mask layer that covers the topmost copper traces, we generate a design file containing pads from all layers, but no traces. This layer is cut in vinyl. While for other layers designers transfer the pads and traces, for the mask layer they peel and transfer the surrounding "background" shape with sensors and obstacles cut out (see Figure 12, left).

### Routing Pads to the Touch Controller

Midas employs an auto-routing algorithm to generate conductive traces connecting electrodes to the Midas touch controller. User-defined obstacles are avoided. We implement Lee's breadth-first maze routing algorithm for single layer paths [15]. For 2D pads, we perform two independent routings: one for the row layer and one for the column layer. Our current algorithm does not generate vias (connections between different conductive layers). When auto-routing 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, though there are designs that cannot be successfully routed. For such cases, the algorithm could be replaced with more sophisticated routing techniques that include user input, though such techniques require that the user has a correct mental model of the routing process.

Midas currently offers generic suggestions when routing fails, e.g.: "Sensor *reply* may be too close to sensor *delete*. Try moving sensors away from the edge and each other."

### Fabrication

Midas generates vector graphics files in SVG format for the electrode and mask layers. These files can 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 35cm bed, cost $200); and copper foil costs a few dollars per foot. Second, copper has excellent conductivity. Third,

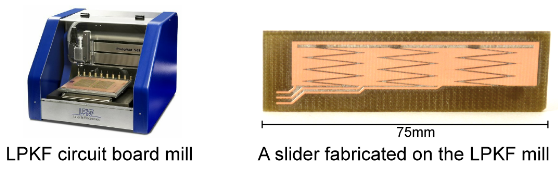


**Figure 8**: Example of a touch sensor manufactured through an alternative process of circuit board milling.

flexible, adhesive foil is easy to apply to non-planar surfaces. However, there are important drawbacks as well. Most importantly, the cutting process and manual weeding (removing background material) determines a minimum feature size for traces. Thin geometric features can also break during transfer, and the weeding process can be tedious 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 touch interactions on conductive surfaces (e.g., aluminum casing) as 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 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 over USB. It recalibrates the touch sensing chips periodically to ensure floating sensor values do not lead to erroneous touch readings.

### Event Output

Once the user has assigned interface scripts to sensors, Midas listens for events from the touch controller. When a touch event matches the key of a saved interaction, that interaction's associated script is executed.

*Record-And-Replay.* In record-and-replay, the user selects a sensor and records a sequence of mouse and keyboard actions that should be played back when the sensor is touched. Early prototypes of Midas used Sikuli for this purpose—a scripting language based on computer vision analysis of screenshots [34]. While more robust 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 [14] to captures and replay both mouse clicks and keyboard events. We share this approach with the BOXES system [13].

*WebSocket Communication with Web Applications.* Record-and-replay is restricted to applications running on the same machine as the sensor editor, and it is limited to mouse and keyboard event injection. To surmount this limitation, Midas can also export touch events to remote clients via a built-in server using the WebSockets API. For example, an application running on a smart phone can open a WebSocket connection to Midas and receive a callback for any Midas button, slider or pad event. The callback function receives an event object describing which sensor changed state, and the value of the new state (e.g., on/off, or 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, is supported by modern web browsers. This means 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 with 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 Apple's iPhone or iPad.

With our WebSockets architecture (Figure 10), designers open a browser and enter the URL of an HTML file they have placed in the Midas server directory. This file opens a

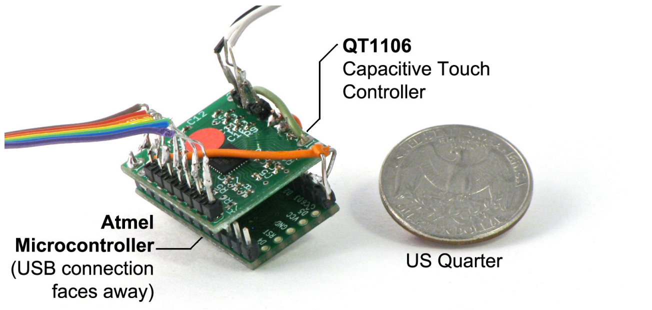


**Figure 9**: The Midas touch controller board uses a commercial capacitive charge transfer detection chip to sense touch events. Events are relayed to a computer via a mini USB connection on the back. The ribbon cables are used to connect to the end of routed traces. A US quarter is shown as a size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

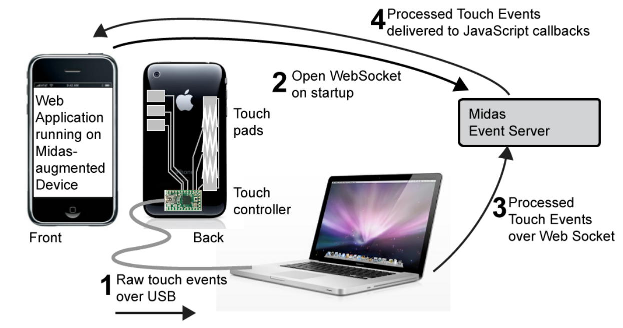

4 Processed Touch Events delivered to JavaScript callbacks
2 Open WebSocket on startup
Web Application running on Midas-augmented Device
Touch pads
Midas Event Server
Touch controller
Front
Back
3 Processed Touch Events over Web Socket
1 Raw touch events over USB

**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.

socket connection from the phone browser to Midas. When Midas receives events from the touch controller, it forwards them to the client, which can then show visual feedback.

*Debugging.* Midas offers basic debugging support. The interface displays incoming touch information from the microcontroller, highlighting activated sensors. We have also implemented basic regular expression filtering on the touch stream to help the user identify potential problems. A timestamp and code representing each touch event is stored in a stream. When a sensor is "on" for more than 10 seconds, Midas reports that that sensor may be touching another wire. When a touch sensor "flickers" on and off more than twice within 500ms, Midas suggests that there may be a faulty connection from that sensor to the microcontroller.

## 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. Our cutter also has difficulties with acute angles such as those in the interdigitated slider. A higher-quality cutter could mitigate these problems.

Second, the touch sensing chips have limited capacity. In addition, continuous sliders use different hardware resources than other inputs and therefore need to be treated differently by the designer. The QT1106 has 7 discrete sensing channels and can support one continuous slider; the MPR121 has 13 discrete channels. The sensor editor keeps track of resources required by the current design and notifies designers if they exceed the capacity of the touch controller. While we currently do not support using multiple touch controllers for a single project, a future circuit board revision could offer such support.

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.

Fourth, Midas does not offer on-device output; a designer must have access to a screen. Use of WebSockets allows this screen to be on any device connected to the Internet, however future work can address on-device output.

Finally, the sensor editor supports only flat, 2D device models. Some 3D shapes can be flattened and these flat files can be imported. Future work could investigate the creation of a plugin for a CAD tool that would aid designers in creating more complex 3D sensor surfaces.

## 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. We have also released Midas as an open-source toolkit to acquire further usage data.

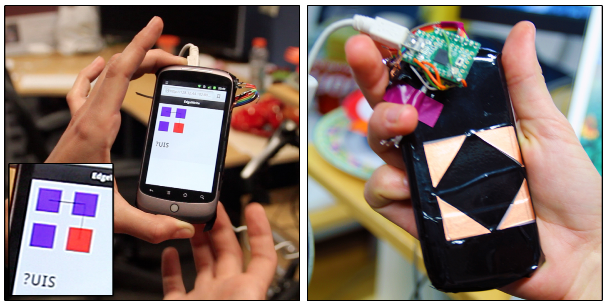


**Figure 11**: We implemented Wobbrock's Edgewrite on the back of a cell phone using a 2x2 pad and WebSocket events sent to a web page.

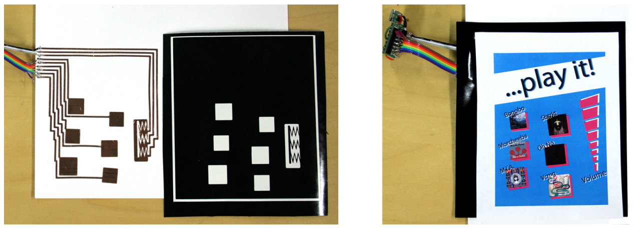


**Figure 12**: Our music postcard lets users sample tracks by different artists. Left: Circuit and mask layer; Right: assembled postcard.

### Example Applications

To demonstrate Midas's expressivity, we built several interactive systems that used all sensor types, and output using both WebSockets and record-and-replay.

*Text Entry.* Wobrrock's EdgeWrite [33] 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 [32]. We cut 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 in future work.

*Game Controller.* To test the responsiveness of Midas's continuous slider, we created a simple game controller for 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 7-inch monitor and mapped slider position to paddle position using record-and-replay. 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

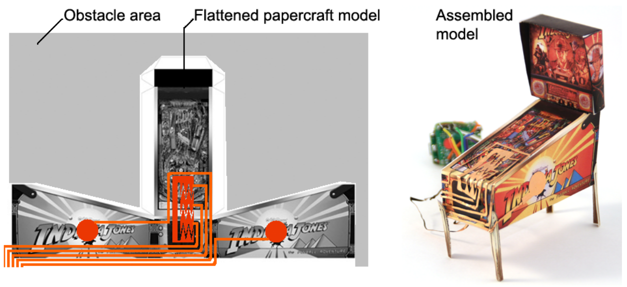


**Figure 13**: For our papercraft pinball machine, we defined the negative space in the template as an obstacle to restrict sensor routing.

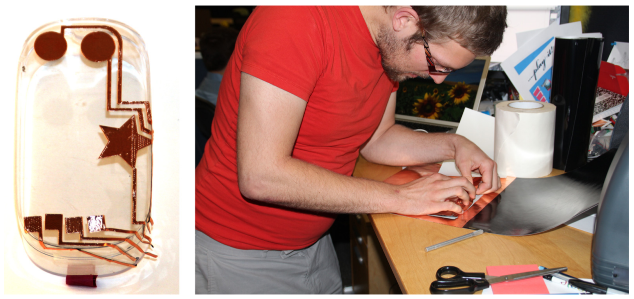


**Figure 14**: A study participant's sensor layout for a PC media player peripheral.

corresponding areas on the poster [22]. We implemented a postcard-sized version of this poster (Figure 12). We scaled back the size 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.

*Papercraft Pinball Machine.* Papercraft is the art of cutting and folding paper into 3D models. To demonstrate how Midas can be used for attaching sensors to more complex 3D shapes, we created a papercraft pinball machine which can control a pinball game on a laptop. First we downloaded a template design for a pinball machine [24], cut it out, and assembled it. We then loaded the already-flattened template model into Midas's 2D editor interface and defined the negative space around the model as a routing obstacle (Figure 13). This guaranteed that all trace routing would be on the surface of the model. After attaching the two buttons for the bumpers and a continuous slider for ball release control, we used record and replay to control a desktop pinball game.

## 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 UC Berkeley (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 received a walkthrough of Midas 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 wished to encourage exploration. Sessions lasted up to one hour. Participants completed a post-task questionnaire with open-ended questions on interface usability and workflow utility.

*Results.* All participants successfully designed media players (Figure 14). Participants commented positively on how Midas aided them with the task of physical construction—both by routing connections and through the generated instructions. Record-and-replay was easy to comprehend and effective for the given task. Though the task did not require programming, two participants expressed interest 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., weeding extra material around the printed sensors). Two desired videos rather than static images. We have since added animations to our instructions to address their request.

Midas's routing algorithm did not find a solution for one version of one participant's design. The participant was unable to identify the underlying problem since he was unfamiliar with auto-routing and the interface did not provide information about the algorithm. In the revised interface we include tips for users on how to adapt designs for successful routing.

Finally, all participants requested richer feedback in the interface about touch events from the hardware controller. The identified challenges are not fundamental to our architecture and have been addressed in recent design iterations.

## 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 physical realization, Midas auto-routes connections, generates instructions, and offers sensor registration by demonstration. Since generated design files are digital, Midas also enables sharing and replication of prototype designs. Our informal evaluation suggests that working interactive objects can be built with our current implementation and that the interface is accessible for prototyping.

Our future work seeks to address identified limitations and expand into new areas. First, we would like to explore methods to *paint* touch-sensitive areas directly onto 3D CAD models and synthesize pads and routes for more complex objects. Such a function could come as a plugin for modeling software such as SolidWorks. In addition, we are investigating fabrication processes and plan to continue exploring conductive ink printing. We are looking into wireless Midas prototypes to permit testing without a USB tether. Finally, we would like to expand beyond capacitive charge-transfer sensing with a prototyping system that can also assist with other inputs. In general, we are interested in more closely integrating user interface design and digital fabrication tools.

## ACKNOWLEDGMENTS

This material is based upon work supported by the NSF GRFP under grant #201246627 and NSF grant #1154235.

## REFERENCES

1. Eric Akaoka, Tim Ginn, and Roel Vertegaal. DisplayObjects: prototyping functional physical interfaces on 3D styrofoam, paper or cardboard models. In *Proceedings of TEI '10*, pages 49–56. ACM, 2010.
2. Arduino physical computing platform. http://arduino.cc. Accessed: June, 2012.
3. Daniel Avrahami and Scott E. Hudson. Forming interactivity: a tool for rapid prototyping of physical interactive products. In *Proceedings of DIS '02*, pages 141–146. ACM, 2002.
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 CHI '08*, pages 423–432. ACM, 2008.
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 CHI '04*, pages 33–40. ACM, 2004.
8. Saul Greenberg and Chester Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Proceedings of UIST '01*, pages 209–218. ACM, 2001.
9. Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. OmniTouch: wearable multitouch interaction everywhere. In *Proceedings of UIST '11*, pages 441–450. ACM, 2011.
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 CHI '07*, pages 145–154. ACM, 2007.
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 UIST '06*, pages 299–308. ACM, 2006.
12. David Holman and Roel Vertegaal. TactileTape: low-cost touch sensing on curved surfaces. In *Proceedings of UIST '11 Adjunct*, pages 17–18. ACM, 2011.
13. Scott E. Hudson and Jennifer Mankoff. Rapid construction of functioning physical interfaces from cardboard, thumbtacks, tin foil and masking tape. In *Proceedings of UIST '06*, pages 289–298. ACM, 2006.
14. Java AWT Robots API. http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html. Accessed: April, 2012.
15. C. Y. Lee. An algorithm for path connections and its applications. *Electronic Computers, IRE Transactions on*, EC-10(3):346 –365, Sept. 1961.
16. 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 DIS '04*, pages 167–175. ACM, 2004.
17. Blair MacIntyre, Maribeth Gandy, Steven Dow, and Jay David Bolter. DART: a toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of UIST '04*, pages 197–206. ACM, 2004.
18. Nicolai Marquardt and Saul Greenberg. Distributed physical interfaces with shared phidgets. In *Proceedings of TEI '07*, pages 13–20. ACM, 2007.
19. Yuki Mori and Takeo Igarashi. Plushie: an interactive design system for plush toys. *ACM Transactions on Graphics*, 26(3), July 2007.
20. Tek-Jin Nam. Sketch-based rapid prototyping platform for hardware-software integrated interactive products. In *Extended Abstracts of CHI '05*, pages 1689–1692. ACM, 2005.
21. Tek-Jin Nam and Woohun Lee. Integrating hardware and software: augmented reality based prototyping method for digital products. In *Extended Abstracts of CHI '03*, pages 956–957. ACM, 2003.
22. BBC News. Playing pop music via paper posters with conductive ink. http://www.bbc.com/news/technology-17339512, 2012. Accessed: April, 2012.
23. Hal Philipp. Charge transfer sensing. *Sensor Review*, 19(2):96–105, 1999.
24. Pinball Paper Toys. http://www.matica.com/paper-craft-toys/61/Classic-Miniature-Pinball-Table-Paper-Toy-Models.html. Accessed: June, 2012.
25. Eric Rosenbaum, Evelyn Eastmond, and David Mellis. Empowering programmability for tangibles. In *Proceedings of TEI '10*, pages 357–360. ACM, 2010.
26. Greg Saul, Manfred Lau, Jun Mitani, and Takeo Igarashi. SketchChair: an all-in-one chair design system for end users. In *Proceedings of TEI '11*, pages 73–80. ACM, 2011.
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 TEI '07*, pages 49–56. ACM, 2007.
29. Nicolas Villar, James Scott, Steve Hodges, Kerry Hammill, and Collin Miller. .NET Gadgeteer: A platform for custom devices. In *Proceedings of Pervasive '12*, pages 216–233. ACM, 2012.
30. Andrew D. Wilson. Using a depth camera as a touch sensor. In *Proceedings of ITS '10*, pages 69–72. ACM, 2010.
31. Raphael Wimmer and Patrick Baudisch. Modular and deformable touch-sensitive surfaces based on time domain reflectometry. In *Proceedings of UIST'11*, pages 517–526. ACM, 2011.
32. Jacob O. Wobbrock. Capacitive EdgeWrite implementation. http://depts.washington.edu/ewrite/capacitive.html. Accessed: April, 2012.
33. 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 UIST '03*, pages 61–70. ACM, 2003.
34. Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: using GUI screenshots for search and automation. In *Proceedings of UIST '09*, pages 183–192. ACM, 2009.