RetroFab: A Design Tool for Retrofitting Physical Interfaces using Actuators, Sensors and 3D Printing

Raf Ramakers[†*], **Fraser Anderson**[†], **Tovi Grossman**[†], **George Fitzmaurice**[†]
[†]Autodesk Research; [*]Hasselt University – tUL - iMinds
[†]Toronto, ON, Canada; [*]Hasselt, Belgium
[†]{first.last}@autodesk.com; [*]raf.ramakers@uhasselt.be

ABSTRACT
We present RetroFab, an end-to-end design and fabrication environment that allows non-experts to retrofit physical interfaces. Our approach allows for changing the layout and behavior of physical interfaces. Unlike customizing software interfaces, physical interfaces are often challenging to adapt because of their rigidity. With RetroFab, a new physical interface is designed that serves as a proxy interface for the legacy controls that are now operated by actuators. RetroFab makes this concept of retrofitting devices available to non-experts by automatically generating an enclosure structure from an annotated 3D scan. This enclosure structure holds together actuators, sensors as well as components for the redesigned interface. To allow retrofitting a wide variety of legacy devices, the RetroFab design tool comes with a toolkit of 12 components. We demonstrate the versatility and novel opportunities of our approach by retrofitting five domestic objects and exploring their use cases. Preliminary user feedback reports on the experience of retrofitting devices with RetroFab.

Author Keywords
Fabrication; Augmentation; Physical Interface; Reverse Engineering

ACM Classification Keywords
H.5.2 [Information interfaces and presentation]: User Interfaces

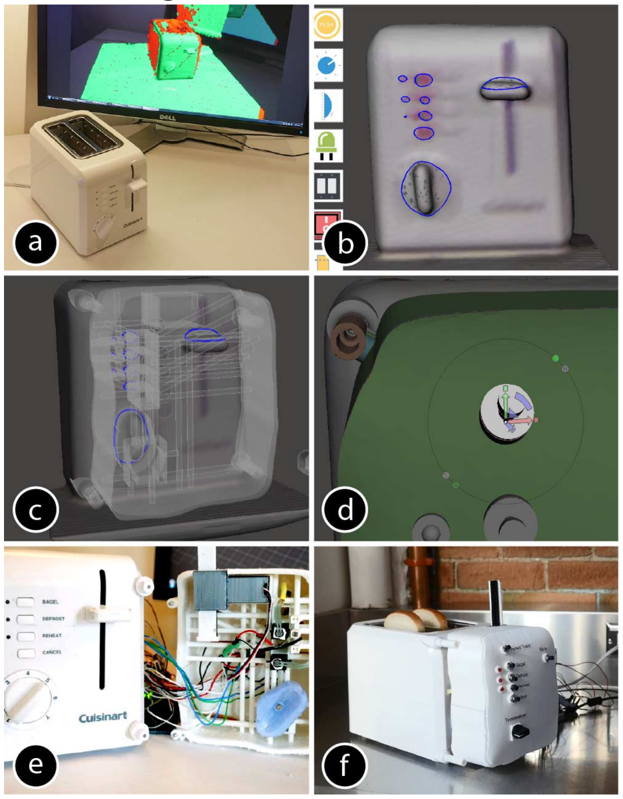
Figure 1: Retrofitting a legacy toaster with RetroFab. (a) The toaster is scanned, (b) the legacy interface is annotated, (c) the attached enclosure is generated, (d) the physical interface and behavior of the retrofit interface is adapted, (e) the enclosure is fabricated and assembled, (f) the new retrofit toaster is perfectly toasting.

INTRODUCTION
Popular computing devices, such as desktops and smartphones are easy to interconnect and their graphical user interface can be adapted to changing user needs. In contrast, devices and appliances such as ovens, thermostats and toasters, are often designed to be static and non-adaptive. Although smart versions of these appliances have become available [30,31] and advancements in sensing technologies for the Internet of Things [12,18] have started to enable basic forms of interconnectivity, they do so at the expense of increased costs or permanent structural changes [4]. Additionally, users have to purchase a new smart device to replace their existing *legacy device* even though it may still be completely functional.

The tangibility and rigidity of these legacy devices make it hard for an end user to change the user interface, as one may do with software applications through plug-ins, reverse engineering [7], or runtime toolkit overloading [10]. For instance, it is not feasible to resolve design mistakes or adapt an interface to users' evolving or custom needs (e.g., impaired users).

To make changes to legacy infrastructures and allow for interconnectivity, one can *retrofit* the physical user interfaces, for example, to augment light switches [32] and dials [33,34]. When retrofitting, a redesigned physical component is placed over top of the original component, thus serving as a proxy interface. Mechanical actuators are often used to manipulate the original interface, while sensors detect states of the device (e.g., via LED indicators). This avoids the complications and risks of fully disassembling and rewiring existing electronic components, and is akin to customizing a software user interface without accessing or modifying its source code [9].

Although these appliance specific retrofitting kits [32-34] are easy to install, the redesigned interface that is now exposed is static and cannot be reconfigured by novices to adapt to personal or changing user needs. Davidoff et al. [8] experimented with customizable retrofit interfaces using the LEGO Mindstorms toolkit. However, the mechanisms had to be manually designed and constructed, required precise structures and brackets that fit over top of the appliances, and did not provide a new physical interface for a user.

To enable users without a technical background in 3D modelling, programming, or electronics to customize and enhance devices and appliances, we present RetroFab, a design tool that automates the process of retrofitting a physical interface (Figure 1). From an annotated 3D scan of an existing legacy object (Figure 1b), RetroFab automatically generates circuitry, firmware and a physical enclosure that precisely fits over top of the legacy interface (Figure 1c-f). These enclosures house mechanical actuators and sensing components to automatically control the device and observe its state (e.g. sensing the state of an LED).

Similar to "smart" versions of appliances [30-31], retrofit interfaces generated with RetroFab allow for automated tasks, interconnected devices, remote control through a companion mobile app, and analytics. Retrofit interfaces designed with RetroFab also go beyond these standard "smart" features by interconnecting multiple heterogeneous devices, and enabling users to customize their layout and behaviors.

The primary contributions of this paper are in the integrated design and fabrication tool that enables custom refactoring of physical interfaces. Specifically, we contribute:

(1) A mechatronic toolkit, consisting of 12 components, optimized for actuating common controls used in household objects and appliances. The toolkit exposes an easy plug-and-play electronics interface to the user.

(2) An end-to-end design and fabrication tool that automatically generates 3D printable enclosures that attach RetroFab toolkit components to legacy devices. Additionally, the tool assists users in the assembly process, as well as the design of new proxy interfaces.

(3) The presentation of a set of sample objects generated using RetroFab, and an exploration of some of the use cases enabled by RetroFab.

Preliminary user feedback assesses the usability and utility of RetroFab, and demonstrates the system's ease of use.

RELATED WORK

This work draws from, and builds upon prior work in 3D model generation, intercepting interactions with legacy objects and design tools for sensor-based interactions. While our work is also related to smart appliances and home automation, a review of such literature is beyond the scope of this work. We refer the reader to existing extensive surveys on such topics [1,4,5,24].

Automating 3D Modeling and Fabrication Processes

The current popularity of 3D printing and digital fabrication has led to a number of recent works in the HCI literature related to modeling and fabrication processes.

Closest to our own work, a number of projects have developed techniques to aid the construction of physical objects that contain electronic or interactive components. Makers' Marks [21] is a system that allows users to create complex physical objects that incorporate elements such as hinges, parting lines and electronics using sculpting material and stickers for annotation. Enclosed [28] is a software tool that enables users to easily generate laser-cuttable enclosures for electronic components. PipeDream [22] automatically routes pipes through 3D models to allow for post hoc insertion of conductive materials. Our work similarly aids the fabrication of objects which can house electronic components, but for the specific purpose of retrofitting an existing physical interface.

We also build upon work that automates, simplifies, or accelerates the design and fabrication process. For example, Fabrickation [16] allows users to rapidly prototype functional 3D objects by integrating LEGO blocks into the design, thus reducing print time. Lau et al. provide a system that decomposes 3D models of furniture into its constituent components and connectors, allowing users to fabricate the discrete elements [14]. RetroFab takes inspiration from this line of research, automating the 3D modeling of the physical enclosure which houses electronic components.

Also related to our work is the recent concept of fabricating directly onto existing objects, or *patching* [25]. This can be accomplished by placing the existing object on a 5 axis rotating platform [25] or on a custom 3D printed support stand [6]. MixFab [29] provides users with a mixed-reality environment where they can incorporate real-world objects into the design of virtual, 3D objects which can then be fabricated, thus making it easier for users to design with existing, tangible objects. RetroFab extends such work by not only printing parts that fit onto existing physical objects, but embedding electronic components which can retrofit an existing object's user interface.

Intercepting Interactions with Legacy Objects

Several researchers have developed novel sensing techniques to provide interaction possibilities to existing objects without dramatically modifying the original object. Touché [20] uses swept-frequency capacitive sensing to detect users' touches and gestures on existing objects. Touch and activate uses a similar approach but uses ultrasonic signals rather than capacitive [17]. Patel et al. [18] describe a method for detecting when appliances are powered on by unobtrusively monitoring electrical noise within the home, opening up possibilities for user-friendly sensing within a smart home. ElectricSense [12] proposes a similar approach, but both of these technologies are limited to sensing and provide no actuation or control capabilities.

Davidoff et al. [8] introduced the idea of mechanical hijacking – using motors to actuate existing controls. Their work explored bespoke designs to actuate specific controls. Recently, there have been several commercial products that capitalize on mechanical hijacking, such as SwitchMate [32], Meld [33], and Lockitron [34]. These devices fit over top of existing light switches, oven knobs, and deadbolts to turn existing controls into smart controls. While user-friendly and easy to install, each device can only interface with a very specific type of control. In contrast to these works, we provide a complete design tool that allows users to easily scan, annotate, and fabricate a new set of controls to retrofit a wide variety of controls.

Design Tools for Sensor-Based Interactions

There are a number of products and research projects that have enabled novices to build electronic devices, such as Phidgets [11], littleBits [3], and .NET Gadgeteer [27]. These plug-and-play devices are easy to use and provide a standard set of compatible components. However, for novice users, it is not always clear which components are required or suitable for a given process, and many of these components still require some programming or electronics knowledge.

Many methods and techniques have been proposed to allow end users to program devices and smart objects more easily. Using simple trigger-action mappings, IFTTT [35] allows users to map events to actions (e.g., a user's GPS location can cause smart light bulbs to turn on or off). While simple, this programming paradigm has been found to be suitable for many of the home automation tasks users desire [26].

Ash et al. [2] describe a visual programming language based on Scratch that allows users to program the functionality of their smart objects. However, this requires the device to already have smart functionality and support for their language. Similarly, Modkit [15] allows users to program Arduino microcontrollers using Scratch and a custom shield, and Jigsaw [13] provides a visual programming language for smart objects. These languages, however, still require users to have some knowledge of programming and electronics components to select the right components.

The Pulsation paradigm proposed in PaperPulse [19] allows for more complex logic definition by supporting linear regression mappings, in addition to mapping discrete triggers and actions. Additionally, Pulsation supports programming by demonstration, allowing for the construction of complex logic rules without programming experience. However, Pulsation was only implemented within the context of paper circuit creation, which we extend to physical user interfaces by centralizing the interpreter on a single PC and adding networking support to facilitate intercommunication.

RETROFAB

Overview and Definitions

The key idea behind this work is to refactor physical interfaces by mounting a redesigned proxy interface over top of the existing form factor that is able to intercept user input and redirect it to the original object using mechanical actuators, while also intercepting device output and redirecting it to the user (Figure 2).

We define the *legacy interface* as the target object which the user wishes to modify. The legacy interface consists of one or more components: *legacy controls* for input (e.g., buttons) and *legacy indicators* for output (e.g., LEDs).

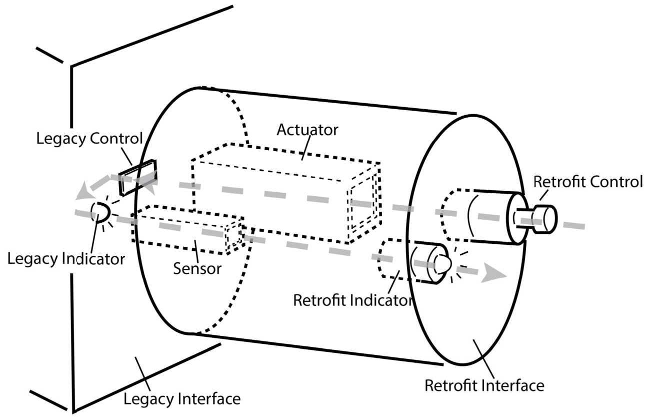

Figure 2: Overview of the retrofitting process. Sensors and actuators are placed on the legacy interface, with new controls and indicators placed on a new, retrofit interface.

RetroFab automatically generates a 3D enclosure structure from an annotated 3D scan of the legacy interface. The system allows the user to define *retrofit controls* and *retrofit indicators* as the input and output components that make up the new *retrofit interface*. A layer of *actuators* and *sensors* are used to interface between the retrofit and legacy interfaces.

Walkthrough: Refactoring a Toaster

The following walkthrough illustrates the process of retrofitting a legacy interface using RetroFab (Figure 1). We use the example of a toaster, and in later sections describe additional functionality and use cases. For the toaster, the legacy interface is composed of the various buttons (cancel, bagel, defrost, and reheat), a dial for temperature control, a lever to push the toast up and down, and a set of indication LEDs.

The following example shows how RetroFab can be used to intercept all interactions and add one extra button on the toaster as a shortcut to a preferred setting – a defrost cycle to thaw the bread, then a mild toasting.

Step 1: 3D Scanning and Annotating Controls

The user starts by 3D scanning the toaster using the Skanect 3D scanning software[1] and a Microsoft Kinect. Before the scan, the buttons and LEDs are highlighted by covering them with tape to ensure their visibility even in low quality 3D scans by novices (Figure 1a). The user loads the 3D model in the RetroFab design tool and annotates the position of the legacy controls and indicators using the brushes in the toolbar on the left. The system contains one brush for every type of supported legacy control and indicator (Figure 1b).

Step 2: Automated Enclosure Design

Once the annotations are finished, RetroFab positions the housings for all actuators and sensors: linear actuators for the pushbuttons and lever, stepper motors for dials and light sensors for LEDs (Figure 3a). RetroFab highlights the region that will be redesigned, and thus covered by the new physical enclosure. The user extends this region to include the area where the new shortcut button will be positioned (Figure 3b).

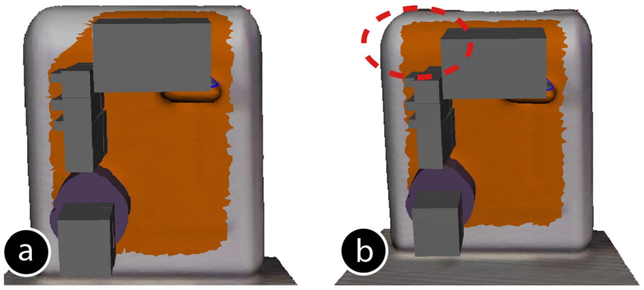

Figure 3: (a) The orange region depicts the enclosure region that RetroFab derives based on the specified components. (b) The user manually extends the region to have additional space for the new shortcut button.

RetroFab responds by generating the 3D model of the enclosure (Figure 1c). Finally, the user specifies the preferred location of the mounting brackets (using a brush) that attaches the enclosure structure to the toaster (Figure 4).

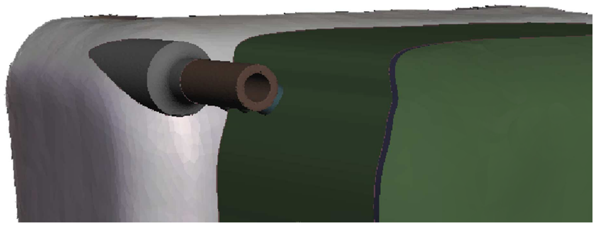

Figure 4: The mounting brackets allow the enclosure to easily be added to or removed from a legacy object.

Step 3: Redesigning the Interface and Behavior

RetroFab automatically integrates a retrofit interface in the front panel of the enclosure structure that serves as a proxy for the legacy interface. If desired, the user can redesign this front panel by repositioning the retrofit components or by replacing them with alternative components available in the toolbar. Figure 1d shows an additional pushbutton being added to the toaster that will serve as a shortcut to the user's favorite toast setting.

By default, retrofit controls are configured to mirror the actions of their associated legacy controls: pushing the new retrofit defrost button on the enclosure structure will cause the legacy defrost button on the toaster to be pushed. Similarly, output is redirected from the toaster to the enclosure structure: when the legacy LED representing the defrost state on the toaster turns on, the corresponding retrofit LED on the enclosure lights up.

The user can alter this default behavior or add extra logic for new components using a *Programming by Demonstration* paradigm [19]. Users demonstrate actions directly on top of the 3D models of the respective retrofit components and can also record functional relationships between these actions. Figure 5 shows the user specifying the logic of the new button on the toaster: If pressed, the lever goes down, and the defrost button is pushed. Additionally, the user can specify that when defrosting is finished (sensed by the LED next to the defrost button), the toast goes back in for a second time at a higher temperature, resulting in perfectly crisp toast!

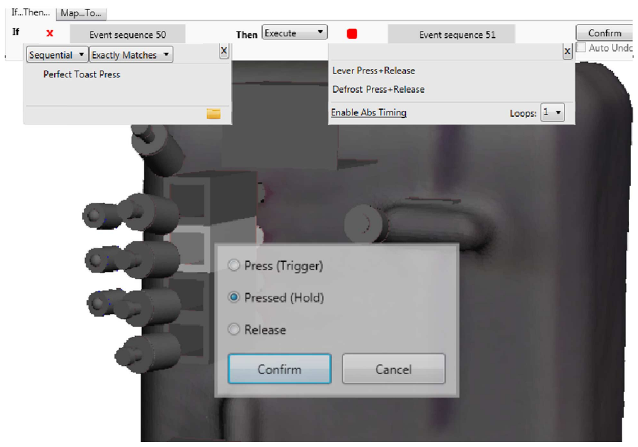

Figure 5: The user manually specifies a logic rule using the Pulsation engine to program the perfect toast setting.

Step 4: Fabrication and Assembly

When the design is complete, RetroFab generates: (1) STL files to be printed out using a 3D printer (FDM printer for our prototypes), (2) Microcontroller code that can be directly uploaded to an Arduino platform, and (3) Assembly instructions to guide users in connecting the actuators and sensors to the microcontroller.

Figure 1e shows the assembled 3D printed enclosure and its embedded actuators, sensors, and retrofit components. Finally, the enclosure structure is attached to the toaster by gluing the feet of the mounting brackets to the appliance (Figure 1f). The mounting bracket is designed so that the enclosure structure is easy to detach using screws, leaving only the feet of the mounting brackets behind (Figure 1e).

[1] http://skanect.occipital.com

Once the enclosure structure is attached, RetroFab guides the user through a process to calibrate all the actuators and sensors.

Step 5: Deployment

All logic defined using RetroFab runs on a central PC which communicates continuously to the retrofit interface. This makes it possible to reconfigure the behavior of devices at run time and allow for interconnectivity. Figure 6 shows the user linking the "snooze" button of his retrofitted alarm clock to his personal "perfect toast" shortcut button on the toaster.

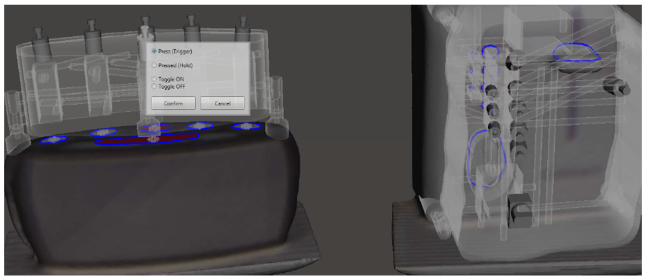

Figure 6: As all of a user's retrofit objects share the same workspace, they can be interconnected to link their functionality and enable home automation tasks.

When the user launches the companion RetroFab mobile application, all retrofitted devices are automatically loaded, and display their functionalities to allow for remote control (Figure 7).

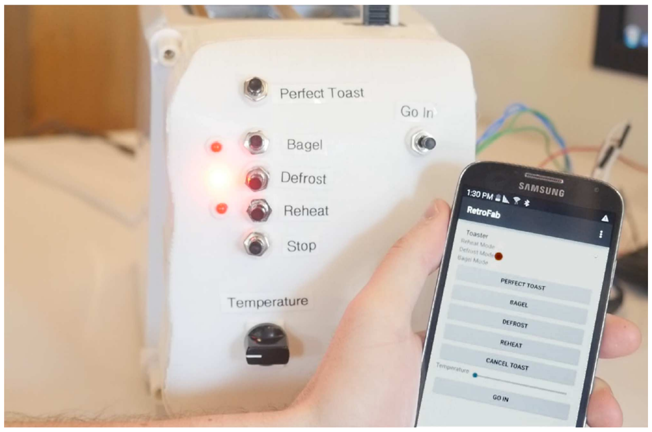

Figure 7: All of the functionality of a retrofit interface is also available in the companion Android application.

RetroFab Toolkit

To enable retrofitting a wide variety of devices, RetroFab comes with a set of electrical and mechanical primitives to retrofit common physical interface components, such as pushbuttons, rotary dials, rocker and wall switches, and LEDs.

Figure 8 shows the full RetroFab toolkit, consisting of (a) actuators, (b) sensors, (c) controls, and (d) indicators. The actuators and sensors are positioned inside the enclosure and concealed, while the controls and indicators are positioned on the outside of the surface, forming the retrofit interface.

For every component in the toolkit, the RetroFab design tool has a specific component housing that is integrated in the enclosure structure to facilitate assembly and ensure precise positioning. This is particularly important for the actuators, which operate the underlying legacy controls.

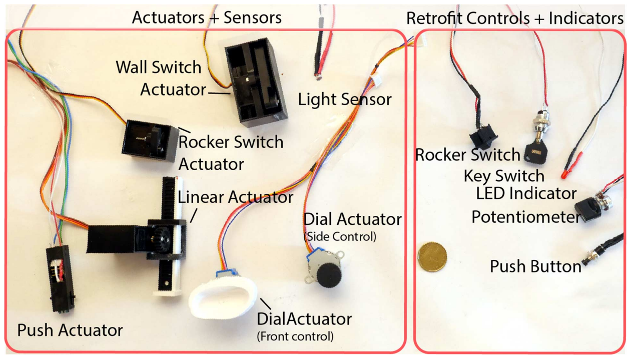

Figure 8: The RetroFab Toolkit; Left) actuators and sensors; the white material of the dial and linear actuator is the parametric component of the design which conforms to the scanned control. Right) controls and indicators.

To make it possible for users without electronics knowledge to use the RetroFab toolkit, wires have a color coding scheme and integrate the necessary electronic components, such as resistors, in them. Our toolkit is easiest to deploy using the Adafruit Motor Shield[2], which avoids complex H-bridge electronic constructions. As such, components are connected directly to the microcontroller by following instructions provided in the RetroFab design tool, avoiding the need for complex electronic wiring designs on breadboards.

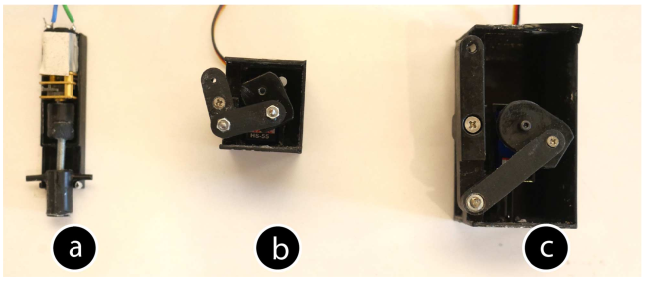

Figure 9: Inside view of a) push actuator, b) wall switch actuator and c) rocker switch actuator.

Our custom designed mechanical actuators use off-the-shelf DC, servo, and stepper motors, in combination with 3D printed transmission mechanisms, to achieve the desired mechanical movement. Figure 9a shows how the pushbutton actuator uses a threaded rod to convert rotational movement of a geared DC motor to linear movement. A pressure sensor is attached to the tip of the piston to reverse the motor when the pressure on the push button reaches the calibrated value. In contrast, the rocker switch and wall switch actuators use micro servos and eccentric crank mechanisms (Figure 9b-c).

[2] https://www.adafruit.com/products/1438

The components that make up legacy interfaces come in different shapes and sizes and are often closely packed together on control panels. Our actuators are therefore designed to be as small as possible while still having sufficient force. The width of our pushbutton actuator is 12mm (Figure 8), making it even possible to actuate legacy controls that have that same distance between their center points. Additionally, actuators are represented in RetroFab by parametric 3D models, to allow them to scale to different sizes and shapes (white material in Figure 8). RetroFab adjusts the size of the rack used in the linear actuator, for example, using information of the 3D scanned model. Similarly, the parameters of the rotary dial actuator allow it to take on the exact inverse shape of the knob of the dial in the 3D scanned model.

To measure properties of controls that are not visible in the 3D scanned model, such as movement range of pushbuttons and dials or brightness range of indicator LEDs, RetroFab employs a calibration procedure. Once the user connects the (temporary) calibration button to the microcontroller of the retrofit interface after the fabrication and assembly phase, the states of each actuator/sensor are calibrated one by one. During this procedure, actuators/sensors are activated and the user presses the calibration button when the actuator/sensor is in the requested state e.g. on/off state for light sensors observing indicator LEDs, min/max state or discrete states for rotary dial actuators. The RetroFab UI supports removing calibration samples and averaging multiple samples.

Enclosures

RetroFab supports the design of two types of enclosures: *attached enclosures*, which attach directly to the legacy interface and *remote enclosures*, which can optionally house the retrofit interface separately from the attached enclosure. Below the design considerations for these two types of enclosures are outlined. Once designed, the method for specifying their behaviors are equivalent.

Attached Enclosures

Attached enclosures are computationally designed with RetroFab and always consist of three layers that are printed separately (Figure 11): (a) the feet of the mounting brackets, (b) the back structure, and (c) the front panel.

The front panel of the enclosure structure consists of component housings for attaching the retrofit controls and indicators that define the new retrofit interface that is exposed to end-users (Figure 11).

The back structure holds housings for components in place inside the structure to precisely position the RetroFab actuators and sensors. Figure 11 shows how rigid support structures connect component housings inside the enclosure design to the outside structure.

Last, the mounting brackets fit the curvature of the legacy interface to ensure a sturdy connection. Figure 10 shows how enclosure structures designed with RetroFab fit on devices with different surface curvatures.

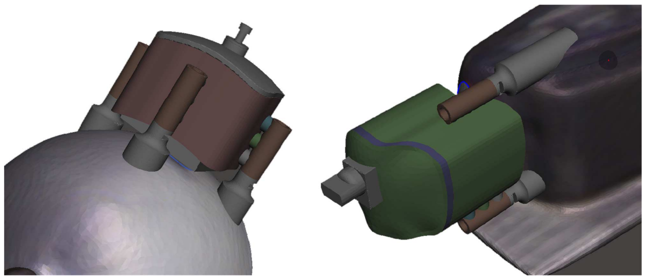
Figure 10: left) Mounting feet conform to the curved surface of a desk lamp, right) the retrofit dial allows for greater positioning accuracy when tuning the frequency of an alarm clock radio.

This layered approach facilitates the assembly of retrofit components on the front panel and back structure which are later glued together. In contrast, the back structure is mounted on top of the feet of the mounting brackets using screws. This is enabled by the T-slot design inside the mounting brackets (Figure 11). As a result, only the feet of the mounting brackets need to be glued to the legacy device. Afterwards, the enclosure structure can be easily removed using the screws, leaving only the feet of the mounting brackets behind (Figure 1e).

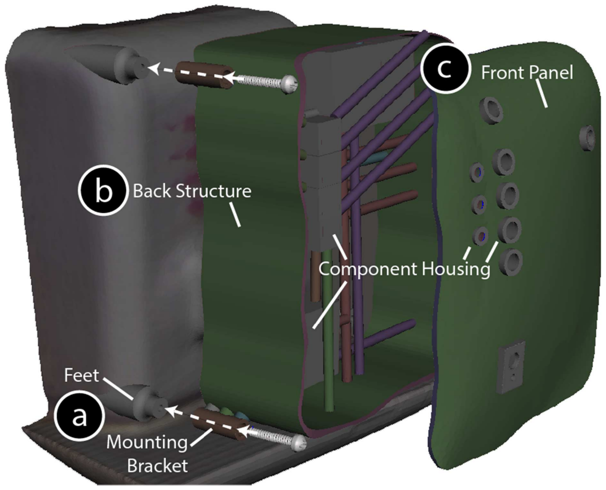

Figure 11: Exploded view of a RetroFab attached enclosure. The mounting bracket (a) is bolted to the mounting feet, which are glued to the legacy device. The back structure (b) holds the motors and sensors which interact with the new retrofit interface on the the front panel (c).

While adding the mounting brackets to the design, RetroFab leaves a gap (approximately 1 cm) between the legacy interface and the enclosure structure. This gap makes it easier to mount the enclosure structure and allows the end-user to observe the state of legacy components while calibrating the actuators and sensors.

Remote Enclosures

RetroFab also allows users to optionally construct a remote enclosure that is not mounted over top of the legacy interface. These types of enclosures only consist of retrofit controls and indicators and communicate wirelessly via a central PC to actuators and sensors that are inside one or multiple attached enclosures. Remote enclosures can be used to design remote controls, reposition an interface to a more convenient location, or introduce new controls to an environment.

To design a remote enclosure, the user loads in any hollow 3D model and adds RetroFab controls and indicators to the front panel. RetroFab responds by integrating component housings in the 3D model that will hold the retrofit controls and indicators in place. Once the design of the remote enclosure structure is finished, the user specifies the behavior between the new retrofit interface and the actuators and sensors in the associated attached enclosures.

Logic and Intercommunication

Besides redesigning the physical interface of legacy devices, RetroFab can also change the behavior of devices. By default, retrofit components on the front panel of the attached enclosure structure mirror all actions to RetroFab actuators behind them: controlling a RetroFab push button or dial on the front panel causes similar actions on the original controls using the RetroFab actuators behind these controls. Similarly, output is redirected from the legacy indicators to the enclosure structure using RetroFab sensors: a light sensor, observing the state of an LED on the legacy device redirects this state to an LED on the front panel.

The user can alter this default behavior, add extra logic, or define logic of additional RetroFab components that were added using the Pulsation programming by demonstration technique [19]. With Pulsation, users define logic between electronic components by demonstrating actions on top of the respective components and record functional relationships between these actions. The pulsation interpreter supports both causal relationships (if-then rules) as well as linear regressions (map-to rules).

In contrast to PaperPulse [19], where the Pulsation interpreter runs independently on every microcontroller, here the interpreter is modified to run on a central logic module (i.e. Windows PC or microcontroller supporting .NET MF), making intercommunication an inherent part of RetroFab. The individual Arduino microcontrollers that control the enclosure structures run a generic firmware that handles the GPIO pins as well as the wireless communication. Even for retrofitted devices that do not intercommunicate, user input and sensor data from the retrofitted interface is first transmitted from the Arduino microcontroller to the central PC. This module then decides to turn on specific RetroFab actuators and sensors, controlled by the same or a different Arduino microcontroller. This approach makes it possible to change the behavior and interconnect retrofitted devices even after the design and fabrication is completed. Multiple independent logic modules can be deployed to avoid single points of failure.

IMPLEMENTATION

The RetroFab design tool is implemented using .NET/C# and builds on the Meshmixer 3D modeling program [23]. The companion mobile application was developed in Java for the Android platform.

Computationally Generated Enclosure Designs

To attach enclosure structures, the automated design process starts with a 3D scanned model that has user annotated regions, specifying the type and position of legacy controls. RetroFab loads and positions component housings related to the annotated controls on top of the scanned model. When components are closely packed together, RetroFab mitigates overlaps between these housings by optimizing their orientation. During this process, the system rotates the intersecting housings one by one, around the normal vector of the annotated region, until all overlaps are resolved. When no solution is found or the process is interrupted by the user, the housings can be manually repositioned

Once the housings of all components inside the enclosure structure are correctly positioned (Figure 12a), RetroFab generates the enclosure design. To support legacy interfaces with different surface curvatures (Figure 10), an enclosure structure is created by extruding the surface region of the 3D scanned model, thus preserving its curvature. The average orientation of the RetroFab actuators and sensors defines the direction of extrusion. Defining the minimal surface region for the extrusion involves the following steps. First, the bounding box of the housing for every component is projected onto the surface along the extrusion direction (Figure 12b). Second, the surface curvature between each component is sampled, resulting in another set of vertices. Together with the vertices calculated in the first step, a mesh of the convex hull is calculated using OpenSCAD[3]. All faces inside this convex hull define the minimal surface region to be extruded to enclose the housings for all RetroFab actuators and sensors (Figure 12c).

When components are located on different sides of the legacy interface, the minimal surface region required for the extrusion can increase substantially (Figure 13a). In these situations, the user can decide to have a separate attached enclosure structure for some components (Figure 13b).

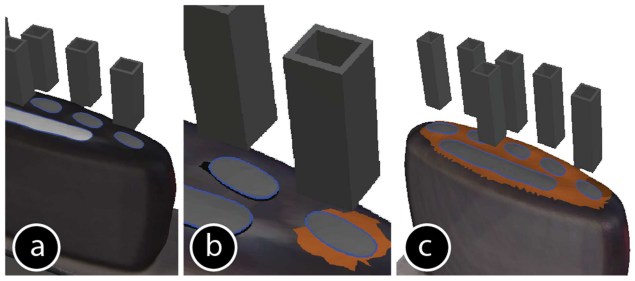

Figure 12: Computing the minimal surface region for the extrusion of an enclosure model. (a) The component housings, (b) the projected surface region underneath components, (c) the final minimum surface region.

[3] http://www.openscad.org

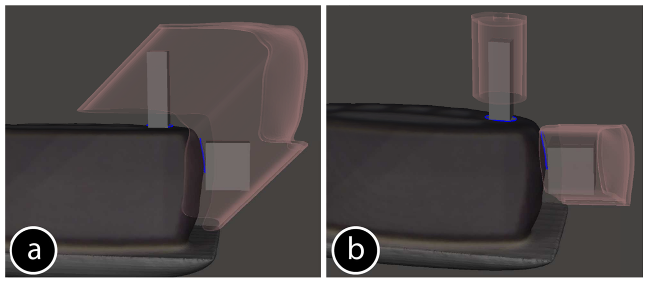

Figure 13: Depending on the surface region covered by the enclosure structure, users can decide to (a) combine actuators in a single enclosure, or (b) group some of them in a separate enclosure structure.

Once the final region for extrusion is defined, it is smoothed and enlarged with 3 mm to account for the thickness of the walls. This surface region is then extracted to a new mesh which serves as the front panel of the enclosure structure (thickness of the front panel; Figure 11). In another copy of this mesh, only the faces on the outermost 3 mm are preserved, resulting in a ring-like shape that will serve as the side panel of the enclosure structure after the faces are extruded (Figure 11). The wall thickness (3mm) was determined through iterative testing. Combined with the support structure that hold actuators in place, these walls provide enough stability during actuation. Thicker walls could cover undesired regions of the legacy interface or make the enclosure heavy and prone to tipping

To connect the component housings to the enclosure structure, RetroFab casts rays from predefined support locations on the component housings towards the enclosure structure. When a valid intersection is found, a cylinder shaped support structure is created (Figure 11).

After the user specifies the locations of the mounting brackets, that connect the retrofit interface to the legacy interface, a predesigned mounting bracket is put in place. To ensure that the feet of the mounting brackets matches the surface curvatures, the Boolean difference is taken between the faces of the mounting bracket and the 3D model, resulting in the removal of all faces that are in inside the 3D model (Figure 14).

In contrast to the housing the components inside the enclosure structure that require a support structure to hold them in place, housings of the components in the front panel are supported by the front panel itself. A Boolean difference operation between the front panel and all the components creates the necessary holes in the front panel (Figure 11).

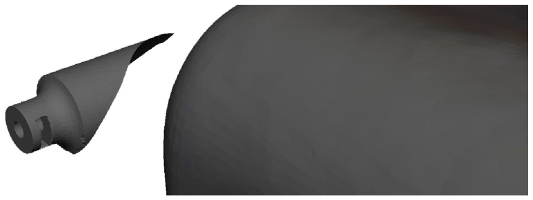

Figure 14: The mounting feet fit the curvature of the legacy object by using a Boolean difference operation with the scanned model.

Since remote enclosure structures only consist of retrofit controls and indicators, only the steps described in the paragraph above are needed to integrate component housings within the custom 3D model.

Parametric Component Designs

For components that consist of parametric parts (white material parts in Figure 8), additional steps are required. RetroFab uses a plane cut to trim the track of the linear actuator to a length that is manually specified by the user (range of movement). For rotatory dial actuators, additional extrusions and plane cuts are applied to create an adaptor that has the inverse shape of the knob of the legacy dial. This approach can also handle dials with an off-centered knob successfully.

Communication with Microcontroller

The automatic instantiation of the generic Arduino firmware on the microcontrollers requires an automatic assignment of control pins. The control pins of RetroFab components can often be connected to multiple pins on an Arduino. If binary output suffices, a digital pin can sometimes be used in place of an analog pin. The system takes this into account and first uses the specified behavior to assign a set of valid control pins to every component. Next, the algorithm selects those pins that maximize the number of components that can be connected given the limited set of pins on the Arduino microcontroller.

Once pin assignments are finished, the central PC communicates the type of components that are used and the pins they connect to the microcontroller. The microcontroller then responds by instantiating code for controlling these components. Afterwards, updates on components' states are communicated to the central PC over XBee or using a wired serial connection.

Communication with the Companion Mobile Application

The mobile application communicates to the central PC using Wi-Fi. Once connected, all retrofit controls and indicators present are transmitted to the mobile device. The companion RetroFab mobile application then automatically instantiates the necessary GUI elements for controlling those components and compiles everything into a single user interface.

EXAMPLE DESIGNS AND USE CASES

Using the RetroFab design tool, 5 legacy interfaces were retrofitted: (a) a wall switch, exposing a rocker switch on the retrofitted interface, (b) A lamp, converting a legacy rocker switch into a push button, (c) the toaster discussed in the walkthrough, (d) an alarm clock with buttons for setting the time (i.e., hours + minutes), setting alarms, and a snooze button, and (e) a stove with a retrofit remote control containing 2 dials and an indicator LED notifying the user when the heating element is warm. Below we discuss a number of use cases that these example design illustrate.

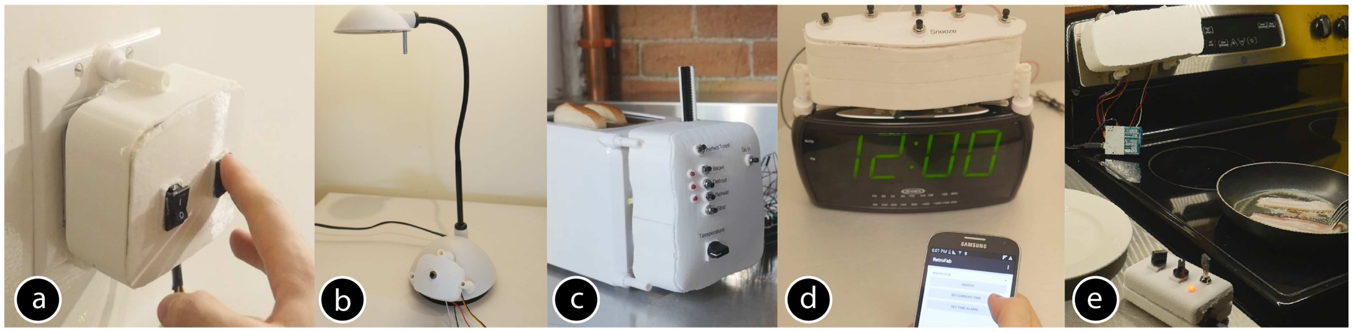

Figure 15: Example retrofit interfaces created using RetroFab: (a) two wall switches, (b) a desk lamp, (c) a toaster, (d) an alarm clock with companion Android application, (e) an oven with remote enclosure.

Remote Interactions

Every retrofit interface created by the user is available through the RetroFab mobile application. This makes it possible to control devices and appliances remotely, such as the light switch when one forgets to turn off the lights. A retrofit interface can also serve as a remote for another retrofit interface. Turning the lamp off when going to bed, turns off the lighting in the room as well, using the retrofitted wall switches.

Locking Out Controls

Digital, as well as physical remotes, make it easy to hide potentially hazardous controls for children. The remote control for the stove can be relocated to a more secure area, or protected further using a key lock (Figure 15e). At the same time, the attached control on the oven contains no physical interface, making it impossible to operate without the remote control.

Resolving Design Flaws and Frustrations

RetroFab also facilitates the process of resolving poor design decisions found in physical interfaces. Controlling the dials located on the back panel of the stove requires moving one's arm over a number of elements, which could have pots or frying pans on them. RetroFab allows for repositioning these controls to a more convenient or safe location, such as the side panel of the stove.

Setting the time and alarm on an ordinary alarm clock (Figure 15d) is often tiring. By retrofitting the interface using RetroFab, a shortcut can be designed for automatically setting the current time after the lock is unplugged or a power outage occurs. This is done by instructing the actuators to press and hold the hour and minute buttons for a calibrated time interval, to increase the time from the known 12:00 start position to the current time.

Shortcuts for Frequently Used or Personalized Actions

As highlighted in the walkthrough, the retrofit interface for the toaster can integrate a personalized button for automatically toasting bread to one's favorite toast settings. Similarly, by retrofitting different wall switches in the home, one can make new buttons that serve as shortcuts for different lighting settings.

Facilitating Interactions for Users with Special Needs

People with disabilities are often unable to operate controls that are found on most devices, as they require considerable amounts of force or fall outside the range of motion they are capable of. The retrofitted desk lamp illustrates how a rocker switch can be converted to a lower force control, such as a push button. A similar push button is used in the retrofit interface of the toaster to replace the heavy mechanical lever.

Statistics on Appliance Usage

Since RetroFab intercepts interactions for every retrofitted control, actions can be tracked and visualized in real time on a fine-grained level (Figure 16). Using this information, statistics over longer periods of time can be compiled to give, for example, data on how often someone presses the snooze button on their alarm clock.

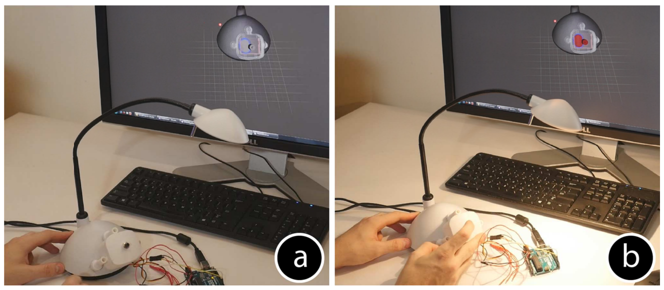

Figure 16: Real-time monitoring of the RetroFit appliance state from the PC.

PRELIMINARY USER FEEDBACK

To understand the experience of working with RetroFab, an informal guided design session was conducted with four participants. Two participants (P1, P2) were experienced CAD users, while the other two (P3, P4) had only limited experience with 3D modelling. P1 had extensive experience in electronic circuit designs, whereas P2 and P3's knowledge was limited to basic prototyping with Arduino, and P4 had no experience with electronic circuits. Each session lasted for approximately 45 minutes.

Participants were first introduced to the concept of retrofitting legacy devices. Then, the participants were introduced to the RetroFab design tool using the example of the retrofitted wall switch (Figure 15a). Once they understood the different concepts, participants were instructed to retrofit the desk lamp using RetroFab (Figure 15b). Due of time restrictions, the generated enclosure structure was 3D printed beforehand and was given to the user during the assembly phase, after they successfully designed their own retrofit enclosure structure. Participants then assembled the 3D printed objects and the electronic circuit by following instructions on the screen. Finally, they deployed the retrofitted desk lamp and controlled it from the RetroFab mobile application. Participants reported their experience with RetroFab through a questionnaire.

All participants were able to retrofit the desk lamp in less than 25 minutes and saw clear benefits in using RetroFab. Participants perceived the entire process as enjoyable and were satisfied with the end result. They reported that the outcome met their expectations.

Three participants (P1, P2, P3) felt they could design a working prototype without using RetroFab, however, they all agreed it would involve multiple iterations and span multiple days. All participants appreciated the straightforward, step-by-step process of RetroFab. They indicated RetroFab would be very useful to them for retrofitting legacy devices in the future. P4 highlighted that RetroFab was an enabling technology for him as he would not know how to retrofit devices without the tool.

P1, P2 and P4 mentioned they are looking forward to see how future versions of RetroFab allow for more customization of generated enclosure structures, such as embedding the enclosure structure design inside a 3D model of choice or giving a retrofitted object a cartoon-like appearance. At the same time, these participants noted that precise placement of RetroFab components inherently allow for anthropomorphism, e.g. making a smiley face with RetroFab components.

Participants recognized that this approach would be useful in different situations, such as controlling the heating at home remotely and saving energy, or for adapting interfaces for impaired users. They all indicated that they would consider deploying this technology at home.

LIMITATIONS AND FUTURE WORK

RetroFab has three important limitations, which we hope can be addressed in future work:

First, RetroFab only generates attached enclosure structures when there is space on the 3D scanned model for attaching the structure. For instance, very small controls are not supported, such as lamps that have small rocker switches integrated in the power cable. To retrofit these devices, future versions could support wraparound enclosure structures that entirely enclose these kind of controls.

Second, the RetroFab toolkit currently supports actuators optimized for operating basic controls used in appliances. In the future, multiple actuators could be developed of various shapes and sizes to reduce size and cost and provide an optimal actuator for each use case. Bigger and more powerful actuators would, for example, allow for retrofitting heavy duty mechanical controls, such as the linear actuator already supported, controls used in industrial machines, and handles to adjust car seats. Another interesting direction for future research is the support of more high-fidelity sensors, such as microphones and cameras, besides the light sensor that is already supported. Cameras and image processing techniques, could allow retrofitting more complex legacy interfaces that communicate states using displays.

Last, the current implementation of RetroFab requires actuators and sensors to be positioned directly in front of legacy controls. In the future, advanced transmission mechanisms could be supported to relocate the actuators out of sight, behind the legacy device, in order to improve the aesthetic appearance of enclosure structures. One could imagine using a single actuator to activate multiple controls to make the retrofit interface smaller. Besides this, the aesthetic appearance could be improved by allowing the user to remodel the enclosure design.

CONCLUSION

As home automation and the development of smart objects continues to rise in popularity, users will desire additional functionality from their existing objects. RetroFab is able to augment these legacy infrastructures using a simple workflow. Many people could benefit from retrofitting interfaces, most prominently members of the maker community, IoT-developers, and researchers. One particularly interesting target audience for retrofit devices are caregivers for disabled or elderly individuals. Retrofitting could allow people with disabilities regain independence and operate legacy interfaces they would otherwise be unable to. Retrofit objects can be interconnected, allowing for simple remote control or automation, or can be used to suit a users' individual needs. The discussion throughout this work shows there are many potential use cases yet to explore, and many opportunities for future work.

ACKNOWLEDGEMENTS

We thank Ryan Schmidt for providing technical advice on Meshmixer, Madeline Gannon for helping with the 3D prints, and Kris Luyten and Jo Vermeulen for their feedback.

REFERENCES

1. Frances K. Aldrich. 2003. Smart Homes: Past, Present and Future. In Harper, R. (ed.) *Inside the Smart Home*, 17–39.
2. Jordan Ash, Monica Babes, Gal Cohen, et al. Scratchable devices: user-friendly programming for household appliances. In *Proc. HCII'11*, 137–146.
3. Ayah Bdeir, and Paul Rothman. Electronics as material: littleBits. In *Proc. TEI'12*. 371-374.
4. A.J. Bernheim Brush, Bongshin Lee, Ratul Mahajan, et al. Home Automation in the Wild: Challenges and Opportunities. In *Proc. CHI'11*, 2115–2124.

5. Marie Chan, Eric Campo, Daniel Estève, and Jean-Yves Fourniols. 2009. Smart homes — Current features and future perspectives. *Maturitas* 64, 2: 90–97.
6. Anthony Chen, Stelian Coros, Jennifer Mankoff, and Scott Hudson E. Encore: 3D Printed Augmentation of Everyday Objects with Printed-Over, Affixed and Interlocked Attachments. In *Proc. UIST'15, 73-82*.
7. E.J. Chikofsky and II J.H. Cross. 1990. Reverse engineering and design recovery: a taxonomy. *IEEE Software* 7, 1: 13–17.
8. Scott Davidoff, Nicolas Villar, Alex S. Taylor, and Shahram Izadi. Mechanical Hijacking: How Robots Can Accelerate UbiComp Deployments. In *Proc. Ubicomp'11*, ACM, 267–270.
9. Morgan Dixon and James Fogart. Prefab: Implementing Advanced Behaviors Using Pixel-based Reverse Engineering of Interface Structure. In *Proc. CHI'10,* 1525–1534.
10. James R. Eagan, Michel Beaudouin-Lafon, and Wendy E. Mackay. Cracking the Cocoa Nut: User Interface Programming at Runtime. In *Proc. UIST'11*, 225–234.
11. S. Greenberg and C. Fitchett. Phidgets: easy development of physical interfaces through physical widgets. In *Proc. UIST'01*, 209–218.
12. Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. ElectriSense: Single-point Sensing Using EMI for Electrical Event Detection and Classification in the Home. In *Proc. Ubicomp'10*, 139–148.
13. Jan Humble, Andy Crabtree, Terry Hemmings, et al. "Playing with the Bits" User-Configuration of Ubiquitous Domestic Environments. In *Proc. UbiComp'03,* 256–263.
14. Manfred Lau, Akira Ohgawara, Jun Mitani, et al. Converting 3D Furniture Models to Fabricatable Parts and Connectors. In. *Proc. SIGGRAPH'11*, 85:1–85:6.
15. Amon Millner and Edward Baafi. 2011. Modkit: blending and extending approachable platforms for creating computer programs and interactive objects. In *Proc. IDC'11*, 250–253.
16. Stefanie Mueller, Tobias Mohr, Kerstin Guenther, et al. faBrickation: Fast 3D Printing of Functional Objects by Integrating Construction Kit Building Blocks. In *Proc. CHI'14*, 3827–3834.
17. Makoto Ono, Buntarou Shizuki, and Jiro Tanaka. Touch & Activate: Adding Interactivity to Existing Objects Using Active Acoustic Sensing. *In Proc. UIST'13*, 31–40.
18. Shwetak N. Patel, Thomas Robertson, Julie A. Kientz, et al. At the Flick of a Switch: Detecting and Classifying Unique Electrical Events on the Residential Power Line. In *Proc. Ubicomp'07*, 271–288.
19. Raf Ramakers, Kashyap Todi, and Kris Luyten. PaperPulse: An Integrated Approach for Embedding Electronics in Paper Designs. In *Proc. CHI'15*, 2457–2466.
20. Munehiko Sato, Ivan Poupyrev, and Chris Harrison. Touché: Enhancing Touch Interaction on Humans, Screens, Liquids, and Everyday Objects. In *Proc UIST'12*, 483–492.
21. Valkyrie Savage, Sean Follmer, Jingyi Li, et al. Makers' Marks: Physical Markup for Designing and Fabricating Functional Objects. *In Proc. UIST'15, 103-108.*
22. Valkyrie Savage, Ryan Schmidt, Tovi Grossman, et al. A Series of Tubes: Adding Interactivity to 3D Prints Using Internal Pipes. In *Proc. UIST'14, 3-12.*
23. Ryan Schmidt and Karan Singh. Meshmixer: An Interface for Rapid Mesh Composition. *SIGGRAPH'10 Talks*, 6:1–6:1.
24. Liyanage C. De Silva, Chamin Morikawa, and Iskandar M. Petra. 2012. State of the art of smart homes. *Engineering Applications of Artificial Intelligence* 25, 7: 1313–1321.
25. Alexander Teibrich, Stefanie Mueller, Francois Guimbretiere, et al. Patching Physical Objects. In *Proc. UIST'15, 83-91.*
26. Blase Ur, Elyse McManus, Melwyn Pak Yong Ho, et al. Practical Trigger-action Programming in the Smart Home. In *Proc. CHI'14,* 803–812.
27. Nicolas Villar, James Scott, and Steve Hodges. Prototyping with microsoft .net gadgeteer. In Proc. TEI'11, 377-380.
28. Christian Weichel, Manfred Lau, and Hans Gellersen. Enclosed: A Component-centric Interface for Designing Prototype Enclosures. In *Proc. TEI'13,* 215–218.
29. Christian Weichel, Manfred Lau, David Kim, et al. MixFab: A Mixed-reality Environment for Personal Fabrication. In *Proc. CHI'14*, 3855–3864.
30. Home | Nest. Retrieved September 22, 2015 from https://nest.com
31. Smart Toast | Breville. Retrieved September 22, 2015 from http://www.breville.ca/smart-toaster.html
32. Switchmate. Retrieved September 22, 2015 from http://switchmate.net
33. Meld | The future of cooking. Retrieved September 22, 2015 from http://www.meldhome.com
34. Lockitron. Retrieved September 22, 2015 from https://lockitron.com
35. Connect the apps you love - IFTTT. Retrieved September 16, 2015 from https://ifttt.com