An Introduction to Developing Handibot Apps [v01]

I. Hello Handibot !

If you've seen our new Handibot, you will appreciate it is capable of doing some amazing cutting, drilling, carving, and machining … all in a nifty little, portable, power-tool package.

But we're going to need a lot of help developing the Apps and Accessories that will enable Handibots for the wide range of people who are going to use of them and for all the various jobs they will be doing. And, we figure it's going to take at least 2 types of contributors: Those with creative ideas for Apps and Accessories, and developers who can help make them happen (and probably a few people who can do both). We'd like to encourage all types of contributors to the broad Handibot ecosystem. This document is our first pass at laying out how it will all work-- defining some of the concepts, and describing how to get started and involved. While we are only providing an early sketch of structures and organization, we hope it will provide you with some suggestions on getting started with Handibot App development. Whether you just want to contribute some thoughts on how to use the tool, or want to code a cool tool interface or design an accessory, we look forward to your getting involved.

Core Handibot Physical Capabilities

First and foremost, Handibots are a new category of power tool. They are all about bringing amazing capabilities of precision and control "to hand" – to extend an individual's physical creativity and productivity with digital technology – all in an immediate and involving format that is largely un-explored.

Today's Handibots are over-built and over-featured. We intended them that way so that early users could explore a full range of potential uses, with as few limitations as possible. These first Handibots are not meant to be as low cost a small CNC as possible. We do anticipate future optimizations that will reduce costs and maybe even extend some capabilities. But today's Handibots pack a lot of potential into a little box ... all so that we can best figure out how to put them to use.

- A Handibot can cut, drill, carve, or machine in a 150mm x200mm x100mm (6" x8" x4") envelope; it works down through its base to cut material underneath; you take the tool to the material; and the material will typically be held in place by pressure from the tool.

- In its basic physics, a Handibot tool does 3-axis, XYZ motion; but the onboard controller has 6-axis capabilities to enable attachments and accessories that extend the kind of work that can be done; there are also up to 12 digital inputs and outputs and several analog channels that are potentially available for added accessories and features; and the onboard processor will allow a wide range of additional connectivity.

- Fitted with a 1.25hp router; a Handibot's power and rigidity allow it to work wood, plastic, aluminum, foam, other similar materials with accuracy [0.006mm (0.00025") step resolution]; allows fine machining for work such as printed circuit boards.

- Fitted with a drag-knife instead of a router bit, Handibots can cut paper, vinyl, cardboard, and other thin materials; fitted with a diamond drag bit they can etch/engrave/mark glass and metal.

- Handibots already function well as small CNC tools. The "Developers Edition" ships with V-Carve Pro software which provides CAD/CAM design capabilities for a normal CAD > CAM > tool workflow. While we are pleased with this functionality of Handibots as capable little CNC tools, our goal is to eventually arm them with software Apps and hardware Accessories that will make them task- and job-oriented "Smart Tools", able to make any sort of project for anyone a lot easier – making as much functionality as possible available, without the need for the challenging steps of the CAD > CAM workflow.

- Handibot is an "Open Innovation", open source hardware project. The full plans for the current Handibots are available in short-form on the Handibot website and in detail from our GitHub repository. The controlling software for Handibot will also be open source and its organization will be described in this document.

An Early Handibot Prototype

A Few Core "OPEN" Principles

Our intention is that Handibot be a complete "open innovation platform". By that, we mean it will be an evolving project and a manufactured product whose hardware and software specifics are openly shared with a community of users, collaborators, and developers. This approach is a forthright experiment on our part – an expression of our interest and enthusiasm for moving towards more rational, environmentally and technologically appropriate approaches to product development and manufacturing. We support "open-source" as a methodology because we believe it is the best way for small companies, small groups, and individual entrepreneurs to leverage collaborative power to innovate and remain competitive. We believe small companies and distributed manufacturing using digital fabrication technologies offer one of the most attractive scenarios for our common future … and that's why we're committed to evolving Handibot in this manner.

ShopBot® and Handibot® are still a business. While not being a particularly crass or profit-oriented gang, we do want to provide stable and fulfilling jobs for ourselves and those we work with. The tools we develop and produce are real physical stuff which has value. We try to keep our products as affordable as possible, but the production of "stuff" involves real materials and real labor. These are not free. In that context we will try and

responsibly manage the business aspects of Handibot evolution in a way that allows Handibot to viably and realistically progress towards being increasingly useful for all at the lowest possible costs. To re-iterate a point made by many in the open-source software and hardware communities already, we are not about "open" because we like "free beer", though we do, but because we believe in the opportunity for advancement that is created by freely shared information and open collaboration.

II. Conceptualizing Apps for Handibot

What kinds of functions will Handibot Apps have?

We figure that by ourselves we might come up with a few good ideas for how to put Handibots or "Smart Tools" to work – and, we do hope to release a number of general-use Apps – but we believe that the community of users will be a deeper source for creative ideas, and that interested and motivated developers will have a lot more capability to innovate compelling and useful software tools and accessories. That's what the "open" App development process is all about.

Handibot App HQ …

In order to encourage early ideas for everyone to contemplate, we've created a Handibot Apps HQ site to collect creative suggestions for Handibot uses. Suggestions and ideas are already being generated on this interactive site. We're looking for ways to further encourage and reward these contributions, but you can already find a lot of really interesting proposals at HQ to stimulate your thinking about ways to put smart tools to use. Here are just a few:

Basic Apps might include …

- Cut variable size holes, at registered locations, with countersinking, pocketing, etc..
- Cut-off functions, diagonals, joints; lots of sizes, types, materials
- Cut-outs for fittings such as lock-sets and hinges; matched to products via libraries?
- Cut-outs for fittings such as electrical or lighting items or plumbing

Apps that work with ACCESSORIES …

- Functionality can be enhanced by developing registration systems that allow positioning and prompting for projects larger than the Handibot's base work area; such systems might be simple and use manual registration, such as fences and fixtures; functionality might be further extended to automated motion on tracks, rails, or wheels for re-positioning. We have already defined an initial jigging system for manual registration which is available for download and cutting or, for purchase pre-cut.
- Apps might take advantage of a 4^{th} axis for rotary indexing; extended length could be added by supporting registration down the rotating part. We illustrated these opportunities with prototype 4 and 5 axis accessories for Handibots at the Bay Area Maker Faire (2014).

- Vertical work of many of the basic types could be facilitated with support systems such as stands, bracketing, or clamps for positioning against walls or other surfaces (there is an example on our Kickstarter project site).

Apps that work with ADVANCED ACCESSORIES …

- Integration might be done with a vision system for positioning the tool (e.g. Kinect)
- An image system could also be used for digitizing shapes in order to do accurate, on the job-site, fitting of trim or components (e.g. edge trim along a stone fireplace)

Construction Job Site App Ideas …

- Pocketer (cut pockets, half-laps, etc)
- Beveler (Make beveled cuts…can already be done with compound miter saw)
- Curver (cut curves/splines)
- Engraver (v-carve text or clip art into material for labeling or signage)
- Puzzler/Joint-Maker (cut any of a wide variety of joints to connect boards or parts)
- General chop-saw applications; especially where distances or angle are critical and change frequently; fit to a job-site type chop-saw stand; could be enhanced with fixturing or automation for handling piece length
- Lockset and door frame cutouts
- Inlays, counter and floor treatments
- Arches circles and ellipses with some sort of registration/indexing; vision system input?
- The company, Homebuilt (http://www.homebuiltcompany.com), proposes to use Handibots to do onsite fashioning of joints and extra parts for digital home fabrication.

At Home Apps and Projects …

- Construction uses as above but also more general hobby crafting; portability creates opportunities that are not possible with full size, immovable, CNC systems; yet, registration allows working in a much larger area than the tool's base work area.

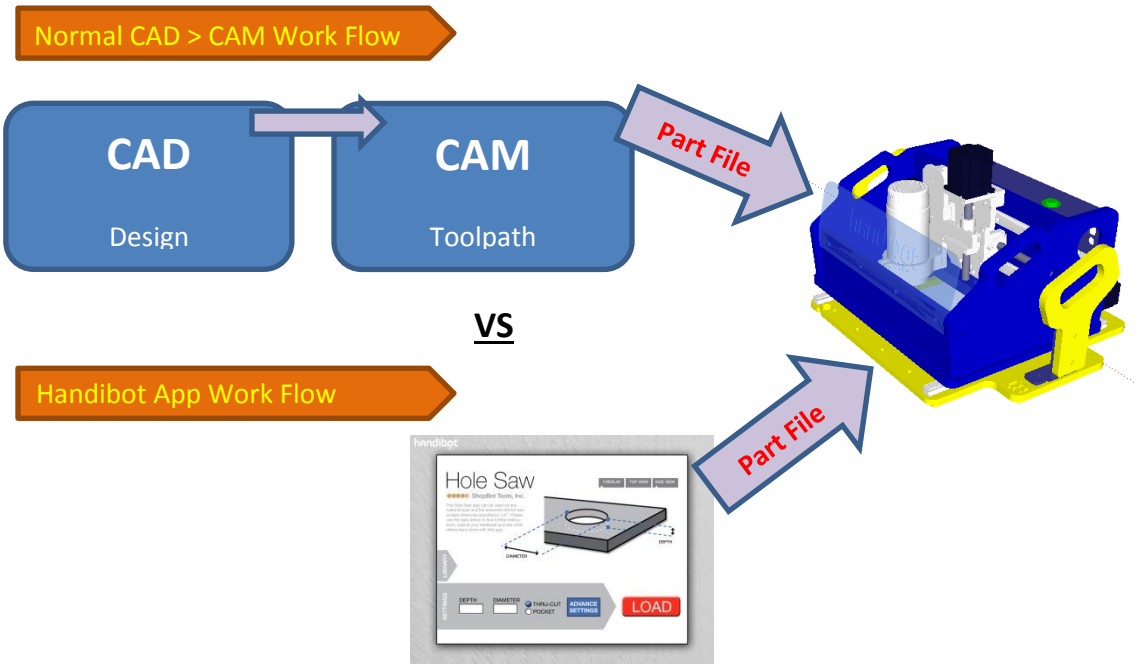
School Apps and Projects …

- A Handibot is a tiny desktop CNC for schools and students of many types … at a school-friendly price; we are getting lots of suggestions for student projects.
- From precise models for theater tech and performance arts to making full size structures, from electronics to sign production.

What An App will really do is pass a motion-describing file …

Fundamentally – a Handibot App will have the function of passing a file of instructions to the Handibot. The file describes the motions that the tool will move through. Such files are variously called part files, tool-path

files, CNC file, and cutting files. They are primarily the list of XYZ coordinates defining the path along which the tool will move in machining a part or making a cut. The file may also contain instructions for controlling or modifying other functionality of the Handibot, such as changing speeds or turning on or off output switches. The file may be flexibly coded in one of several different code formats. This file is the same type of file as would be created in a normal CAD > CAM > CNC Tool workflow.



A Preliminary Categorization of "Handibot Apps"

{Just a short digression on wording: "App", or application, is a term overloaded with meaning. There are many types and levels of software applications involved in running the tools we are talking about, from firmware on the control card to PC software. However, for present purposes it makes a lot of sense for us to speak in terms of a "Handibot App" or a "Smart Tool App" as a light, user-facing, software module that runs tasks, jobs, of projects using a "smart" tool. Here we are using the popularized, contemporary meaning of "app" like a smart-phone module that handles a single job and is widely identified by today's device-using culture as a "portable unit of functionality"—a small software package for accomplishing a task. "App" provides clarity to general users of the task orientation that we have in mind and will be useful for communications in marketing. Typically we will use the full description, "Handibot App", to make it really clear what we are referring to and we will stick with it as our basic term. More specific variations, as below, can be used to describe the specific functionality that will be made available in particular types of Handibot Apps. }

We imagine at least 3 different types of Handibot Apps. These types have different basic purposes, work in somewhat different ways, and are likely to operate from different locations in the Handibot system.

1- **Handibot Utility Apps**; Handibot Apps that carry out job functions such as cutting a hole of a specific size, making a joint of a given size or type, doing some sort of simple marking or lettering, and so forth – doing a basic "job" resulting from the user making a few choices. This type of Handibot App

might be considered a defining type of on-the-job “smart” functionality. Because such Handibot Apps are fundamental ‘tools’ for using a Handibot, they will typically reside on the device that is used to run a particular Handibot or optionally, on the tool itself. Being resident on the device or tool means the Apps will be available offline, always ready for use whether or not there is an internet connection available in the work environment.

2- **Handibot Catalog Apps**; Handibot Apps that make a completely predefined and usually complex cut such as making a cutout for a specific fixture like a sink or faucet set , or, pocketing an area to mount a specific type of hardware. In this case, the Handibot App would carry out the function of looking up the correct cutting file in a library or database, perhaps first reading a QR code on the packaging of the hardware item. An App might also look up shapes or diagrams in a catalog and turn them into cutting files; this might be done, for example, with libraries of moldings or carvings. In this latter case, some scaling might be involved in running the App. But the idea of a “Catalog” App is that this category of functionality primarily represents data retrieval, rather than interaction with the user over choices or parameters, and that developer’s challenges are related to structuring efficient libraries. Because of the library functionality, these Apps and data would typically be available to users from the web, resulting in cutting data that is first downloaded to the device they are using to run the tool, and then passed to the tool.

3- **Handibot Design & Project Apps**; Are ambitious types of Handibot App that will involve creating a design or project rather than handling a single cut. Typically, these Apps will integrate input from the user in customizing the items to be produced. That is, they will be “parametric” in one fashion or another – having user adjustable values or parameters. They might help a user create a jewelry item, a piece of furniture, a house! A Handibot Design & Project App is more complex than a Utility App because this type of Handibot App has 3 substantive components: It involves interacting with the user; it involves the design itself and its parametric customizability; and, it involves the software mechanisms for coding a particular version of a tool-path file for a user. These Apps are likely to reside on the web where they are used by a Handibotter to get a particular instance of the design or project for a specific use.

   For developers, Project Apps thus involve creating a design and determining how its components will be parameterized or customized – as well as presenting it to the user and generating the code from the design that is used to run the Handibot. In the best of all worlds, development of these types of Apps should be less about the coding tasks and more about design. We are working to make App development as easy as possible by providing functions through API calls, library resources, and App templates.

   But even with these resources available, a developer will be handling all 3 aspects of a Design App including being responsible for generating the tool-path code that guides machining. So, we are looking forward to the evolution of more of a “designer friendly” system which would offer an environment for design-oriented developers to create parametric designs and make them ready for delivery as Handibot Apps – with the presentation and tool-path coding being handled relatively automatically. Imagine a furniture designer, for example, who wants to create exciting customizable

furniture but who is not a programmer or interested in the creation of the App interface itself. That designer would like to be able to utilize some type of parametric design tool to help create designs for Handibot users based on methods for arraying parametric components. We hope to create, or encourage, one or more systems of this type – systems that will allow designers to deliver their own customizable projects via the Handibot HQ store without explicitly programming the App user interaction functionality. This is one of our goals in creating an eco-system for delivering a wide range of Handibot Apps to Handibot users.

TABLE 1: The Handibot App Framework TABLE lays out these categories of Apps in terms of how we see them being programmed, where they can be situated, and where we see our own emphasis focused. We'll refer to this table as we move through the subsequent sections on developing and delivering Handibot Apps. Note that this Handibot App Framework is open (and a little vague). There are multiple possible ways to create functionality for Handibots and serve up Apps. The hardware and software structures described below will allow (and usually support) them all. The range of workable options or methods is indicated by the light-purple cells – in principle you can do things just about any way that you choose. The dark purple cells indicate those approaches we at Handibot/ShopBot intend to emphasize as best methods.

III. Developing Apps for Handibot

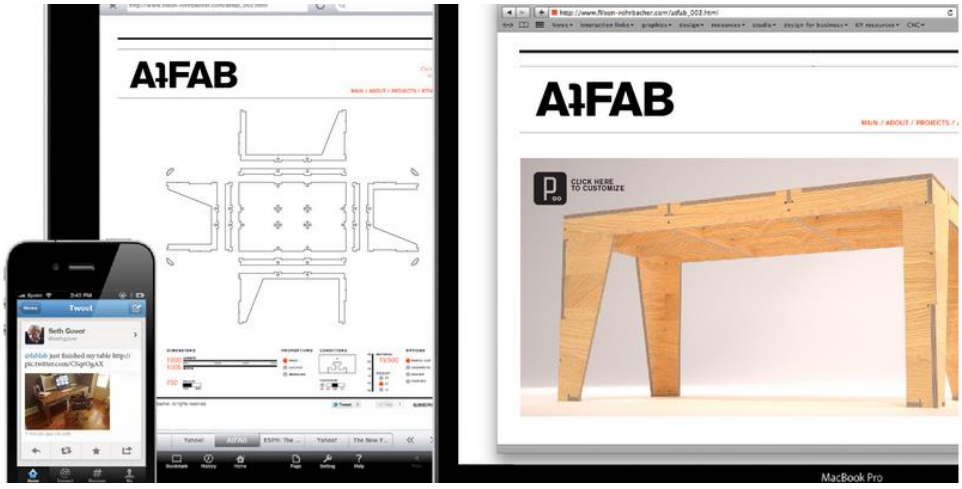
A. The Programming Concept for a Handibot App*

The Handibot App will usually make a tool path …

For the tool user, Handibot Apps will replace the difficult work flow of creating and drawing a shape or cut in a CAD program and then using a CAM program to generate a tool-path for it. Having to master both CAD and CAM is what makes it hard for many people to put CNC to use (of course, learning about cutters and how to hold down material add a bit of challenge too). So, the App will create the tool-path file to send to the Handibot – replacing the difficult CAD/CAM process. The Apps' functionality for the user will generally be to make the process of defining the cutting and machining for a regular job or task, EASIER – easier than engaging in the full CAD/CAM process that is so daunting to many who have considered and fled from CNC – especially when compared to the *apparent* "click to print" ease of 3D printing.

In many cases an App will be "parametric". It will turn a generic need into the specific instructions required for a user's particular situation. This could be for a simple shape such as a hole whose size is parametric, as in a Handibot Utility App; or, it could be for customizing a much more complex Handibot Design & Project App. For example, a Handibot Design App that produces a coffee table might allow the user to first define the height and width of the table. These parameters would be used to modify an underlying coffee table design and output the specific tool-path to make the exact size coffee table that the user wants. These changes could be more complex than simply changing the scale of parts. They might involve things like the shape, size, and placement of joints. The App developer would have done all the heavy lifting to program the generation of the tool-path, with the user making a few choices to customize the table to their own specific needs. Of course, the App might have a very compelling interface that allows the user to preview what the table will look like, maybe with a couple of sliders to manipulate length and width. For example, have a look at a very

sophisticated chair-making-system being developed by our friends at SketchChair -- this level of sophistication may be a bit of overkill for early Handibot Apps, but it is an inspiring project. We believe that eventually it will be compelling Apps like SketchChair that make a Handibot (or any other small digital fab machine) a compelling "Smart Tool".



AtFAB provides another example of parametric furniture design

***We are evolving a NEW Hardware/Software System for Handibots and for ShopBot!**

The hardware/software and communications system being described here is in development. It is not the software or current Control Card provided on the Kickstarter Edition or the "Developer" Edition of Handibots. It will become available with the release of a new Control Card and SBC (single board computer) for Handibots. This new hardware system can be retrofitted to existing Handibots and will be made available to our "early adopter" Handibot owners at cost. The new system builds on the open-source, G2 motion core that will run on our new Control Card.

If you are not yet familiar with our ongoing Control Card development and our transition plans to the G2-based motion control and, you can read about transitioning to a new Control Card, here. The new Control Card can be retro-fitted to all existing Handibots and ShopBots as well as being the core for all of our ShopBot Control Systems going forward. The development board for this project is the Arduino DUE and the final Control Card will use a similar, 32-bit ARM micro-controller as on the DUE. For the Handibot and other "connected" CNC tools, we will be putting a small SBC (e.g. Raspberry Pi, Beagle Bone) in front of the Control Card to handle connectivity and manage files and the streaming of data to G2. This SBC will provide most of the App housekeeping functionality developers will need. The core G2 project is currently accessible on the Synthetos Github site and active development is underway. Our ongoing work on the more general and overarching digital fabrication motion platform (FabMo) can be found at the ShopBot Tools Github site. We are expecting considerable evolution of the G2 and the FabMo systems in the next few months and you may want to hold off getting too involved with them just yet – unless you are really curious, and expect to work at this lower level of programming.

B. Basic App interaction with Handibot

Primary work the passing Tool-Path Files …

The primary function of any Handibot App is to communicate machine motion instructions to the tool. Either of two different file types may be used to communicate cutting and machining instructions to a Handibot (these can be used in combination, as well).

a. Standard (NIST) g-code files; g-code is a universal syntax for communicating with CNC (computer numerically controlled) tools. There are many variations of g-code, some more ugly than others, but Handibot will work with a basic subset of the very common NIST-defined codes. This is true for both the current/original "ShopBot" Control System as well as the g-code for the G2 Motion Control System that will be on future Handibot Control Cards.

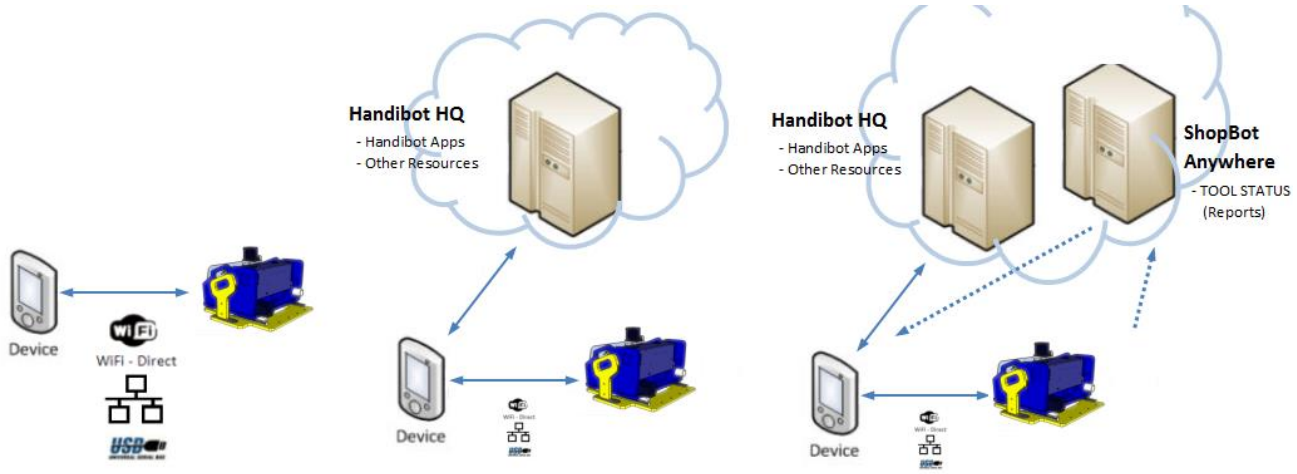
b. OpenSBP code; this is an open-syntax code for controlling digital tools. ShopBot contributed the code format to the CNC community many years ago. It is almost identical in concept to g-code but uses symbols, syntax, mnemonics, and programming extensions that are much easier for humans to read and understand than g-code. Many current CAD/CAM software systems post OpenSBP as well as g-code. OpenSBP is available for the controller on current Handibots and will be supported on the G2 version as well. It adds additional programmability to the standard set of g-code instructions. (www.opensbp.org; with the most recent info found with your current ShopBot Control software install: C:\Program Files (x86)\ShopBot\ShopBot 3\Help\ComRef.pdf and in the Developer Tools folder and handbook: C:\Program Files (x86)\ShopBot\Developer Tools\ProgHand.pdf.

Developers are welcome to use either code for communicating their tool-path or part files to Handibots. And, with a few exceptions, you will even be able to use the two code types interchangeably within the same file. As a general rule, if you are already comfortable with g-code, you should use that. If not, OpenSBP will be easier for you to get going with. See links above for references for each system.

A third and additional method to pass cutting and machining instructions utilizes direct, low level communication with the G2 motion core through an asynchronous JSON-type, command structure. This system will allow a very direct control over the operation of a Handibot (or other CNC) in this open system, including communicating machining instructions (though, in this case, you will need to manage the flow of vectors rather than just provide a file for execution). We don't expect this system will be used for the typical App but is available and may be appropriate for situations where there are extensive ongoing interactions between the App and the Handibot during execution of some job or task and where the developer wishes to take on the streaming of instructions to the low level motion system. See the Synthetos Github site.

The method of communication with the Handibot tool will be via either USB, a Local Network Connection (wired or WiFi), or WiFi-Direct. Optionally, we intend to enable a system (ShopBotAnywhere) for remote monitoring of the status of the tool via cloud access as determined/allowed by the user's hardware options, communications choice, and permissions setup.

Overview of the Methods that will be available for Communicating with Handibots



Primary Local Pairing USB |Local Net | WiFi-Direct	Local Pairing with Link to Handibot App HQ for Apps and resources	Local Pairing, with Link to Handibot App HQ, and optional status tracking via ShopBotAnywhere
- Local Pairing established by "Linker" - File management and other functions using Fabrication API - Safe robust local control of tool	- Access to Apps at Handibot HQ - Full Handibot resources available - Interaction with remote Apps; download of local Apps - Clarifies that HQ interactions are between HQ and Device **- Device is connected to tool locally**	- Optional Status Monitoring through Cloud - Tool state viewable from any Device (with access permission) - No remote activation/control of tool, only info about tool status viewable

C. Deeper into the workings of Handibot Apps

Think of the new hardware/software system for Handibots as a general digital fabrication motion platform (FabMo for short). It is made up of the hardware components including: cloud services, a user device for communicating with the Handibot (such as a smart phone, tablet or PC), and on the Handibot a small, single-board computer (SBC), as well as a Control Card, and CNC Interface Board and related motion hardware.

The software functionality provides for running and managing the files generated by Apps and providing the developer with additional functions and utilities. "**Linker**" software running on the user's device will identify nearby Handibots and securely log in and "pair" with a selected tool. In addition to streaming files, the fabrication "**Engine**" will provide API functions to the developer for managing files and configurations as well as for optionally monitoring the progress, activity, and status of the Handibot. We note that the **"G2 core"** of the motion system will be directly accessible to developers interested in working with hardware at a low level (we do not expect those developing Handibot Apps will typically need or use this access).

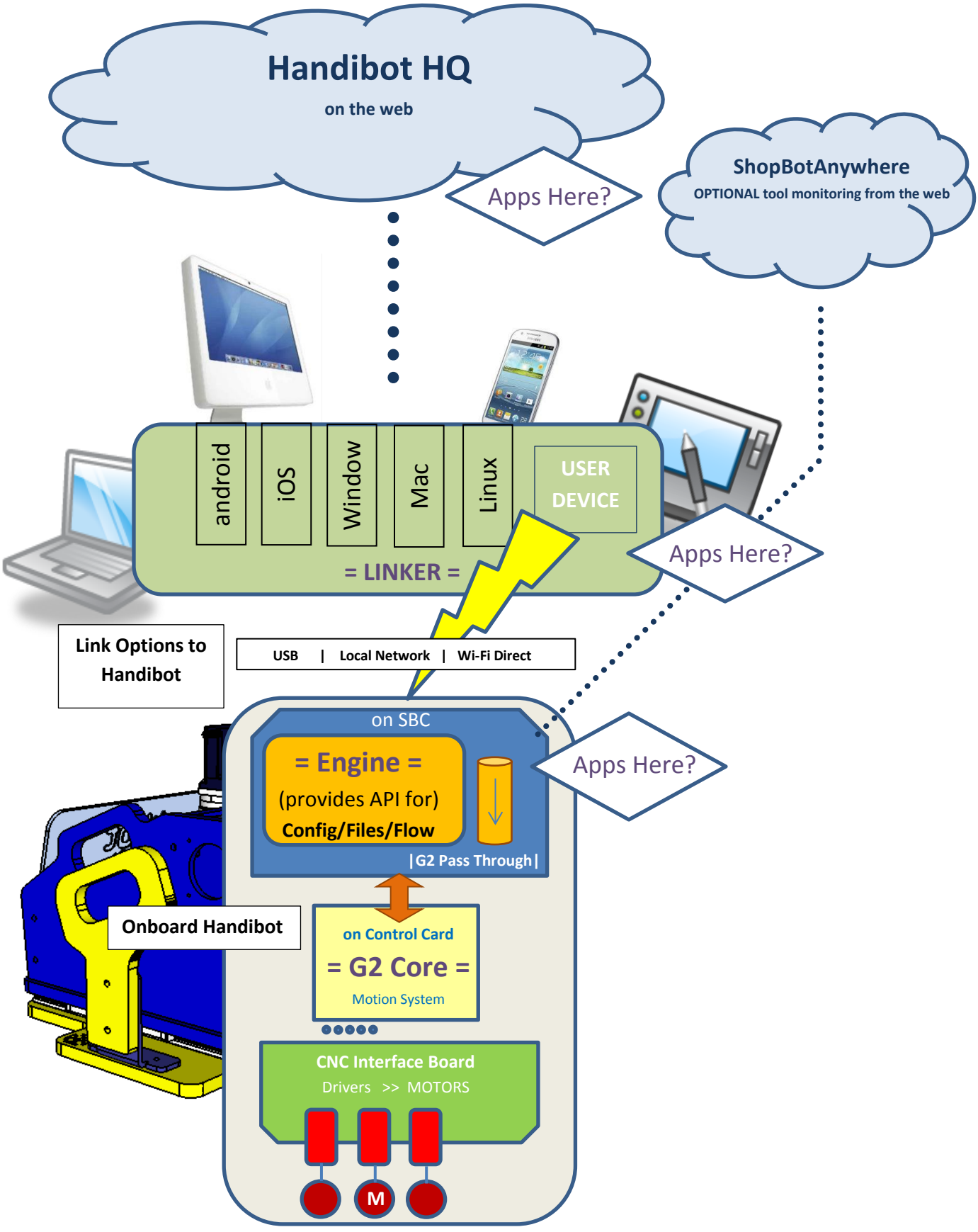
The coming FabMo API …

While Linker (running on the user device) will take care of hooking you up to the tool by whatever means is available, the fabrication API (provided by the fab engine running on the SBC) will manage all interactions with the tool and provide most of the additional tool-manipulation and configuration-functionality you will need. By whatever option your device is connected to a Handibot, the interactions will be the same. Fab API calls are broken down into 4 categories.

- Status & Configuration
- File Management
- File Execution
- Direct Commands

The exact details of the API calls are still a work in progress. But we will begin to share details of the specifics, shortly. Everything defined by the API will be simple data exchanges used to run the tool. All data is JSON formatted. There will be no exchange of "webpages" and the API is not intended to be responsible for any sort of presentation. However, we are also developing a higher level library to further simplify standards tasks as well as templates and examples for how one might receive and present and data. These will be made available as the SDK for the FabMo platform.

Our goal is to provide an arrangement of components that allow users to interact with fabrication tools easily, both locally and remotely, as well as provide structure for developers of a wide range of Handibot Apps. Table 1 reflects our preferences for how Handibot Apps should be instituted because we believe this will help insure all software components work well together and still preserve as much flexibility and functionality as possible, even when running without access to the internet. Additionally we see the Handibot HQ website serving as an important resource for users and for the sale and distribution of Handibot Apps in a simple and secure fashion. We hope to make it a preferred option for you for distribution of Handibot Apps and Accessories.


Handibot HQ
on the web
Apps Here?
ShopBotAnywhere
OPTIONAL tool monitoring from the web
android
iOS
Window
Mac
Linux
USER DEVICE
= LINKER =
Apps Here?
Link Options to Handibot
USB | Local Network | Wi-Fi Direct
on SBC
= Engine =
(provides API for)
Config/Files/Flow
|G2 Pass Through|
Apps Here?
Onboard Handibot
on Control Card
= G2 Core =
Motion System
CNC Interface Board
Drivers >> MOTORS
M

Coding and locating your Handibot Apps…

What language will you use to run your Handibot App? There are two answers to this question. In theory, you can program in any language that you would like that is appropriate for your target device(s). Your App needs to interact with the user to collect information for generating the tool-path file. Then, assuming it has already paired with a Handibot using the Linker, it communicates using the API or higher level resources, passing a tool-path file and doing whatever general management is necessary. These interactions should be possible and straightforward from almost any language.

However, because we imagine that most developers will want to reach as many Handibotters as possible and will not want to develop individual apps for a variety of different devices, we encourage development in a web-oriented language that will run in a browser. Simple Handibot Utility type apps can be deployed to the user's device or tool to make them always available, even if the Handibot only has a local connection. More sophisticated Design & Project Apps can be run from the cloud.

What can you do at this point? As we are finishing up the first round of development in laying out the system, it is a good time to become familiar with the tool-path language you choose for communicating machining instructions to the Handibot. You can also start working on the front-end that provides your interaction with the user, and validate the functionality of your App by sending the generated tool-paths to the tool. We have provided V-Carve Pro with the Developer Edition to help you explore the nature of tool-path files and to illustrate what is involved in the CAM process of converting a design concept into machining. We also thought this was good software just for using Handibots – the software Vectric provides really a solid and straightforward CAD/CAM design system and serves well for most needs when simply using the Handibot as a CNC tool.

Simulating Apps with Today's Hardware …

During our present Handibot development period and with the existing hardware and software on Kickstarter Handibots and "Developer Edition" tools, the App-to-File-to-Handibot flow can be simulated using the Windows-based ShopBot Control System running on a PC. This approach is a bit of a kludge, but does allow simulation of the work flow and a bit of the look and feel. One simply uses the ShopBot software system for passing a file via a Windows command line or via the registry (this functionality is described in: C:\Program Files (x86)\ShopBot\Developer Tools\ProgHand.pdf).

It is not a perfect simulation in terms of appearance, because the ShopBot Sb3 software will be running on the screen at the same time, but it offers a relatively complete functional simulation with calls to the current ShopBot software standing in for the API calls. Our "Hole-Saw" example is a very simple browser-based App that will run locally, coded in JavaScript, but it is based on calling the current Sb3 ShopBot software running on your Handibot (if you don't have a tool, it will work in Preview Mode). You can also use this example App to run the tool remotely, from any device on your local network, see readme after installing the file. Feel free to build your own test cases from this example. Hole-Saw Example, here.

Where will I distribute or sell My App?

It's up to you …
As indicated by the light purple colored regions in the Handibot App Framework Table we intend that Handibots be as open as possible. You will be free to distribute Apps (and run them) however you wish. That includes selling them through existing web stores for specific devices or interest groups, or just giving them to your friends.

But we hope you will consider …
We do have our preferences (the dark purple stuff) and we encourage you to consider distributing/selling your Apps through the Handibot HQ website. This site will be the center of Handibot community activity and a source for Handibot resources for all Handibot users. It will be the natural site for users to come when looking for a Handibot App.

We encourage open-source apps, but we also understand that software development takes time and effort; it is often important that effort should be compensated. Our store will offer you an automated solution for delivering your Apps and receiving payment. It will support a developer gallery to describe and feature your Handibot Apps and to manage reputation ratings so that as the Apps ecosystem develops, users will be able to easily find your Apps and to share their confidence and delight in your work.

At this point, we do not plan to enforce software or encryption locks on delivered Apps or files. We could probably do this based on monitoring schemes in the user's Handibot or device. But, in our experience with CAD/CAM IP over the years, we find the *majority of users play fair* and respect IP when it is made available at reasonable pricing and when the ownership policies and rights are explained. Contorted protection schemes usually just produce user frustration and work to the disadvantage of all.

To the degree that you are able to develop parametric Handibot Design & Project Apps, their advantage is that with cloud delivery they provide to the user only a final tool-path file. Thus, the user is only buying a particular and specific instance of the output of the parametric system (e.g. one size and shape). While it is conceivable to scale files when they are run by a CNC tool, once a project design gets a little complicated scaling will not correctly alter shapes. When priced appropriately so that the user feels good about the project, this custom part-file approach encourages fair play because it means that specific customizations of the project are always best and most easily acquired through the App delivery system itself rather than trying to game it.

Additional Thoughts on Developing for Digital Fab in General …

Our discussion of Apps and Projects here is oriented to Handibots. But, we view these concepts as being appropriate for development of software for all coming Smart Tools and for future digital fabrication tools in general. Thus, though a developer might be concerned that work on Handibot projects addresses a relatively small audience, we imagine that these types of Apps and Projects will have applicability far beyond just Handibots. Indeed, one reason for our part-file focus in Handibot App development rather than an alternative, lower-level machine-specific language is that for the foreseeable future, CNC-style part files are

likely to be the primary way digital fabrication equipment, both subtractive and additive, is run. Your development effort will not be limited to our initial Handibot market – you will be developing for small Smart Tools today, and for extended digital fab markets of the future …

TABLE 1: Handibot App Framework