A Tclish Espresso Machine:
— Project update (after 7 years)

A programmer talks about espresso.

John Buckman in Vienna, Austria (July 2023)
License: Creative Commons by-sa v3

http://decentespresso.com/doc/
In 2019 I presented here, and 13,000 people watched!

https://www.youtube.com/watch?v=Ey30Cg1fM0Q
What is this?
an espresso machine built from scratch
Why?

- because espresso was pre-scientific
- espresso machines were hard wired for one approach only
So? What to do?
sensors were needed to capture real data

https://www.youtube.com/watch?v=AS6FGEWScmU
An app was needed to display data neutrally and truthfully

And here we are. Our basket is saying we're at 90, the scace is saying we're at 89.6, 89.8,
Programming espresso shots

a visual espresso-programming tool to encourage experimentation
Communicating
sharing of learning, best practices, integration of research results

https://www.youtube.com/watch?v=VEzH1JBA3k8

https://visualizer.coffee/

Decent Espresso Profile Generator

https://www.youtube.com/watch?v=3lb63xBNrzw
Simplifying & Sharing
Distribution of what was learnt to non-experts

The 4 mothers: a unified theory of espresso making recipes

I've been working on a "unified theory of espresso making recipes", which results in 4 "mother" recipes.

The optimal espresso curve

But before I plunge into that, I want to make the argument that there is an arguably optimal pressure curve for espresso recipes.

You can see it as the pressure profile that occurs naturally when constant flow water is used to make an Allongé recipe coffee on the Decent. The resulting pressure is a reflection of the declining puck resistance over time, as the puck loses material to the espresso drink.

Coffee-making variables

Some key variables for espresso

Watch our video: Espresso is difficult

Flow rate:
Depending on the type of espresso you make, the flow rates and pressure will vary. For a drink with no milk or very little milk, use a higher flow rate. For a drink with more milk — like a cappuccino or latte — you want a "heavier" espresso, so the flow rate should be slower.

Pressure:
The default 9 bars of pressure often produces a more acidic espresso, such as the classic Italian. Less pressure (lower bars) increases the body and produces a more "chocolatey," richer flavor.

Temperature:
- Light roasts need higher temperatures.
- Medium or dark roasts use lower temperatures.

Each coffee preset has defined settings already. If you want to experiment, you can ask them in Settings.

Feedback on the Decent screen

The Espresso tab on the tablet screen gives you a lot of feedback about your coffee.

What do the readings mean?
On the bar charts, the dotted line represents the goal. The solid line shows what your brew is doing in real time.

On the far right, you’ll see data about your coffee, such as the preinfusion time, pouring time, and weight of final coffee output.

Yes, the settings are editable

How to use Preset and Profile editor (Pressure/Flow rate) page

When you pull your coffee, the tablet displays some of your results.

TIME: An ideal pour time is around 25s-30s, but this cup poured in 20s. Some adjustments could be made to lengthen that time.

WEIGHT: This preset is configured for a 2:1 ratio. Ideally, 18 grams of coffee beans should output 36 grams of coffee (see right side, bottom). In this case, the output was 35.4 grams of coffee (left side, bottom).
Tolerance for imperfection
a UI was needed to display that data neutrally and truthfully
DUI widgets library

by Enrique Bengoechea

https://github.com/decentespresso/de1app/blob/main/documentation/decent_user_interface.md
Native-looking widgets
App Extensions

Extensions

- D_Flow_Espresso_Profile
- DPx_Flow_Calibrator
- DPx_Screen_Saver
- DYE
- Example Plugin
- Hazard customizations
- keyboard_control
- Log DEBUG
- Log Uploader

Visualizer Upload

Username: demo@demo12
Password: passwd

Auto-Upload
Minimum shot seconds to auto-upload: 6
Last upload: Last shot not found

Keyboard Control

- Espresso Key: e
- Steam Key: s
- Hot Water Key: w
- Flush Key: f

Next Step on Espresso or Steam key tap
App Extension Example

```plaintext
set plugin_name "example"

namespace evl plugin:plugin_name {
  # These are shown in the plugin selection page
  variable author "Jiayi"
  variable contact "mail@coffee-mail.de"
  variable version 1.8
  variable description "Minimal plugin to showcase the interface of the plugin / extensions system."
  variable name "Example Plugin"

  proc build_ui () {
    variable settings
    # Unique name per page
    set page_name "plugin_example_page_default"
    # Background image and name button
    add_del_text $page_name 1288 1318 text (translate "Done") .font Helvetica-Bold .fill "#888888" .anchor "center"
    add_del_button $page_name 678 (translate (button)) .fill [settings(sound_button_shape) page_in_html_show_off_extensions] .width 900 1230 1580 1410

    # Option:
    add_del_text $page_name 1288 300 text (translate "Example Plugin") .font Helvetica-Bold .width 1280 .fill "#444444" .anchor "center" .justify "center"

    # The actual content. Here a list of all settings for this plugin
    set content_textfield [add_del_text $page_name 600 388 text ""
    set description [array get settings]
    set description "$description@key: $value"

    .can itemconfigure $content_textfield .text $description

    return $page_name
  }

  proc on_espresso_end (old esp) {
    log text "espresso ended"
  }

  proc on_function_called (call code result op) {
    log text "start_sleep called!"
  }

  # This file will be sourced to display meta-data. Dont put any code into the
  # general scope as there are no guarantees about when it will be run.
  # For security reasons it is highly unlikely you will find the plugin in the
  # official distribution if you are not being run from your main
  # REQUIRES:

  proc main () {
    variable settings
    set settings [namespace current] "Changing settings"
    add_plugin_settings "example"
    msg [namespace current] "Adding settings"
    msg [namespace current] "Registering espresso ending handler"
    register_state_change_handler "Espresso" "Idle" plugin:example:on_espresso_end
    variable description
    msg [namespace current] "Tracing function call"
    trace add execution start_sleep leave plugin:example:on_function_called

    # register gui
    plugins gui example [build_gui]
  }
}
```
Easily make new UIs
skin development via a language-within-a-language approach
Sample Skin

package require del 1.0

# DECENT ESPRESSO EXAMPLE SKIN FOR NEW SKIN DEVELOPERS

# you should replace the JPG graphics in the 2560x1600/ directory with your own graphics.
source "$hom edir]/skins/default/standard_incl udes.tcl"

# the standard behavior when the DE1 is doing something is for tapping anywhere on the screen to stop that. This "source" command does that.
source "$hom edir]/skins/default/standard_stop_buttons.tcl"

# example of loading a custom font (you need to indicate the TTF file and the font size)
#load_font "Northwood High" "$skin_directory]/sample.ttf" 60
#add del_text "off" 1280 500 -text "An Important message" -font {Northwood High} -fill "#2d3046" -anchor "center"

# text and buttons to display when the DE1 is idle

# these 3 text labels are for the three main DE1 functions, and they X,Y coordinates need to be adjusted for your skin graphics
add del_text "off water" 510 1076 -text [translate "WATER"] -font Helv_10 bold -fill "#2d3046" -anchor "center"
add del_text "off steam" 2048 1076 -text [translate "STEAM"] -font Helv_10 bold -fill "#2d3046" -anchor "center"
add del_text "off espresso" 1280 1076 -text [translate "ESPRESSO"] -font Helv_10 bold -fill "#2d3046" -anchor "center"

# these 3 buttons are rectangular areas, where tapping the rectangle causes a major DE1 action (steam/espresso/water)
add del_button "off" "say [translate {water}] $:settings(sound_button_in);start_water" 210 612 800 1416
add del_button "off" "say [translate {steam}] $:settings(sound_button_in);start_steam" 1748 616 2346 1414
add del_button "off" "say [translate {espresso}] $:settings(sound_button_in);start_espresso" 948 584 1606 1444

# these 2 buttons are rectangular areas for putting the machine to sleep or starting settings. Traditionally, tapping one of the corners of the screen puts it to sleep.
add del_button "off" "say [translate {sleep}] $:settings(sound_button_in);start_sleep" 0 0 400 400
add del_button "off" {show_settings} 2000 0 2560 500
Skin writing extensions

Screen Variables are Tk text widgets that refresh

add_del_variable "steam_1" 537 1250 -text "" -font Helv_10_bold -fill $tappable_text_color -anchor "center" -textvariable {[seconds_text $::settings(steam_timeout)]}

Mechanism for any Tk widget

# 3 equal sized charts
add_del_widget "off espresso espresso_1 espresso_2 espresso_3" graph 20 267 {
  bind $widget [platform_button_press] {
    say [translate {zoom}] $::settings(sound_button_in);
An Over-the-air update mechanism
and global challenges with making that actually work
Challenges

• shipping the same app on Android, Windows, OSX and Linux
• converting the Tcl app into a WebApp using mp4 streaming
• what kind of people embraced Tcl and why
• what kind of people hated Tcl, why, and what happened then
• the move from open-source-but-one-programmer to a full open source multi-programmer participation via Github and relinquishing control
• challenges of supporting many different resolutions, tablets and Android versions
• Right-to-left languages
Future

• Surprising findings how Tcl outperforms competing other programs (in other languages) trying to do similar things

• What hasn't turned out well, and what we’re trying to do about it. Bluetooth is our main problem.

• The future of Tcl for us, as Python, Javascript, as others launch competing apps

• Cloud integration

• Two apps at once: point-of-sale and order queue management. Mobile-ordering app. iOS via webapp/mp4.
Other Decent Uses of Tcl & Naviserver

- Customer admin
- Espresso machine customization
- Real-time inventory shopping cart
- Quickbooks API Integration
- Espresso machine shopping
- Internal staff metrics
- Boxing & Shipping via APIs
- Real UPS/Fedex monitoring via APIs