Contents

1 Introduction

2 Abstract

mis mis mis.

3 Disaggregation Algorithm

3.1 Learning New Devices

3.2 Detecting Changes in Steady State

3.3 Device Checking

4 Graphical User Interface

The graphical user interface(GUI) is made up of several layers of code. The setup aims to provide a very flexible set of libraries for defining the appearance of the touch screen.

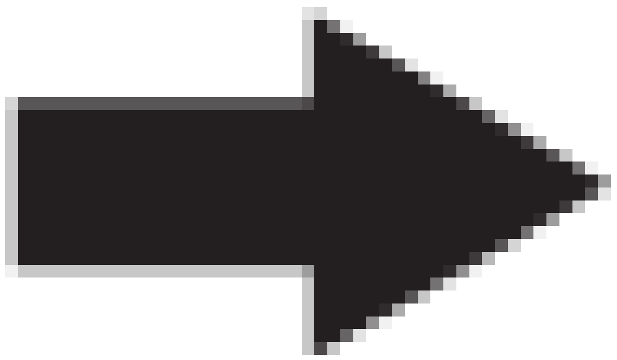

Figure 1: *Phase-correct PWM generation simulation*

4.1 Windows

At the very bottom of the layout hierarchy is so-called Windows. A window in this context is a struct containing only a single char that defines its type. All function that operate on windows take void pointers as arguments. These void pointers can be cast into the appropriate type of pointer by examining the type char.

Several types of windows exist:

- PictureWindow: A rectangular button that draws as a bitmap.
- RectangleWindow: A rectangular button that draws as a rectangle of specified color. Can have a border.
- ProgressBar: Used to indicate that the learning algorithm is running.
- Graph: Plots the readings obtained from the SmartBox in real time.

In these, the most important is by far the RectangleWindow. The RectangleWindow contains a pointer to an onClick function, meaning that each button can easily have a desired function attached to it by calling setOnClick(). Furthermore, the RectangleWindow also contains a char pointer that is used for drawing text in the middle of the rectangle. The text to be drawn can be set using setText().

4.2 Layouts

The Layout struct combines several Windows in order to create a part of the GUI. The Layout library contains functions that allow for drawing of entire layouts, passing touch events on to child windows, and retrieving pointers to particular windows(i.e. for changing the backgroundcolor of a rectanglewindow).

4.3 Animations

In order to allow for easy animations and timing, an animations library has been written. The library uses a timer to update any posted animation every 20ms using an Interrupt Service Routine(ISR).

The basic functionality is:

1. post() called. Initializes the animation to run by saving pointers to the element to animate as well as a function to be called when an update is necessary. This function also takes a starting value and a increment that is applied on every update.
2. ISR runs. This will update the starting value by the increment and call the supplied update function.
3. The update function returns 0, meaning that the animation is not yet done.
4. ISR runs again.
5. The update function returns 1, indicating a successful end of the animation.
6. The animation framework is reset and awaits the next animation to be posted.

4.4 Graphs

5 Web Server

5.1 Storing Data as XML

5.2 JavaScript User Interface

6 Discussion

6.1 Problems

6.2 Further Enhancements

7 Conclusion