```
    1: /*
    2: * File: run.qml
    3: * Description: Main controller/view for the frontend. Root element that
    4: *               instantiates all components.  Lasts the runtime of the GUI.
    5: * Project: Carberry Pi
    6: * Author: Ryan McHugh
    7: * Year: 2020
    8: */
    9:
   10: // root
   11:
   12: import QtQuick 2.11
   13: import QtQuick.Window 2.4
   14: import QtQuick.Controls 2.4
   15: import QtQuick.Extras 1.4
   16: import "./partials"
   17: import "./js/header_back.js" as HeaderBack
   18: import "./js/header_info.js" as HeaderInfo
   19: import "./items" as Items
   20:
   21:
   22:
   23:
   24: ApplicationWindow {
   25:     id: root
   26:     visible: true
   27:     width: 800
   28:     height: 480
   29:     title: "Carberry Pi [Development]"
   30:
   31:     // Variables
   32:     property var style: main.config['style']['current']
   33:     property var header_list: {
   34:       'info': false,
   35:       'text': "",
   36:       'stack': stack,
   37:     }
   38:     color: "white"
   39:
   40:     header: head
   41:
   42:     // Source: https://forum.qt.io/topic/62267/how-we-can-create-2-second-delay-or
-wait-in-qml/7
   43:     Timer {
   44:       id: timer
   45:       running: false
   46:       repeat: false
   47:
   48:       property var callback
   49:
   50:       onTriggered: callback()
   51:     }
   52:
   53:     Timer {
   54:       id: timer_extended
   55:       running: false
   56:       repeat: false
   57:
   58:       property var callback
   59:
   60:       onTriggered: callback()
   61:     }
   62:
   63:     // Javascript Functions
   64:     function setTimeout(callback, delay)
   65:     {
   66:       if(timer.running){
   67:         console.error("nested calls to setTimeout are not supported!");
```

```
   68:       }
   69:       timer.callback = callback;
   70:       timer.interval = delay;
   71:       timer.running = true;
   72:     }
   73:
   74:     function setTimeoutRepeated(callback, delay, cont){
   75:       if(timer.running){
   76:         console.error("nested calls to setTimeoutRepeated are not supported!");
   77:       }
   78:       timer_extended.callback = callback;
   79:       timer_extended.interval = delay;
   80:       timer_extended.repeat = cont;
   81:       timer_extended.running = true;
   82:     }
   83:
   84:     function sendInfo(text){
   85:       header_list['info'] = true;
   86:       header_list['text'] = text;
   87:       headerObj.list = header_list;
   88:       setTimeout(function(){
   89:         header_list['info'] = false;
   90:         headerObj.list = header_list;
   91:       }, 1000)
   92:     }
   93:
   94:
   95:     // run function to return conditional
   96:     function sendInfoExtended(text, run){
   97:       setTimeoutRepeated(function(){
   98:         header_list['info'] = !run();
   99:         header_list['text'] = text;
  100:         headerObj.list = header_list;
  101:       }, 1000, !run())
  102:     }
  103:
  104:     function printTest(){
  105:
  106:       console.log("\n_____TESTING_____\n\n")
  107:
  108:       var test_dict = stack.currentItem.children[0].testValues()
  109:
  110:       console.log("\n___DASH___")
  111:       console.log('Speed: ' + (main.handler['speed'] == test_dict['SPEED'] ? "YES"
 : "No"))
  112:       console.log('RPM: ' + (main.handler['rpm'] == test_dict['RPM'] ? "YES" : "No
"))
  113:       console.log('Coolant: ' + (main.handler['engine_temp'] == test_dict['COOLANT
'] ? "YES" : "No"))
  114:
  115:       console.log("\n___Diagnostics___");
  116:       // console.log(test_dict['diag_1'] == 22 ? "YES" : "No")
  117:       for(var iter = 1; iter <= 6; iter++){
  118:           console.log('temp' + iter + ' ' + (main.diagnostics[test_dict['diag'].ge
t(iter - 1)['key']] == test_dict['diag'].get(iter - 1)['value'] ? "YES" : "No"))
  119:           // console.log(test_dict['diag'].get(0)['value'])
  120:           // console.log(main.diagnostics['temp1'])
  121:       }
  122:       console.log("\n_____END TESTING_____\n\n")
  123:
  124:
  125:     }
  126:
  127:     Item {
  128:       id: head
  129:       Header {
  130:         id: headerObj
  131:         list: header_list
```

```
  132:         context: main
  133:       }
  134:     }
  135:
  136:
  137:
  138:     StackView{
  139:       id: stack
  140:       anchors.fill: parent
  141:       initialItem: view1
  142:       objectName: "stack"
  143:
  144:       onCurrentItemChanged: currentItem.objectName != "" ? sendInfo(currentItem.ob
jectName) : null
  145:
  146:       Transition {
  147:         id: transition_enter
  148:         PropertyAnimation{
  149:           property: "opacity"
  150:           from: 0
  151:           to: 1
  152:           duration: 500
  153:         }
  154:       }
  155:
  156:       Transition {
  157:         id: transition_exit
  158:         PropertyAnimation{
  159:           property: "opacity"
  160:           from: 1
  161:           to: 0
  162:           duration: 500
  163:         }
  164:       }
  165:
  166:       popEnter: transition_enter
  167:       popExit: transition_exit
  168:       pushEnter: transition_enter
  169:       pushExit: transition_exit
  170:
  171:
  172:       signal sig_exit(var exit_code)
  173:       signal sig_restart(var exit_code)
  174:
  175:
  176:
  177:       Component {
  178:         id: view1
  179:         Item {
  180:           id: mainItem
  181:           SwipeView{
  182:             id: swipeView
  183:             anchors.fill: parent
  184:
  185:             currentIndex: 0
  186:
  187:             function testValues(){
  188:               var dict = {
  189:                 'SPEED': dashObj.speedVal,
  190:                 'RPM': dashObj.rpmVal,
  191:                 'COOLANT': dashObj.coolantVal,
  192:                 'diag': diagObj.props,
  193:               }
  194:               return dict
  195:             }
  196:
  197:             onCurrentIndexChanged: function(){
  198:               // console.log(this.currentIndex)
```

```
199:              var text = ""
200:              if(this.currentIndex == 0)
201:                text = firstPage.objectName
202:              else{
203:                text = secondPage.objectName
204:                if(text == "Diagnostics")
205:                  diagObj.props.refreshModel();
206:              }
207:
208:              sendInfo(text);
209:              // HeaderInfo.create(header_list['text'])
210:            }
211:
212:
213:            Item {
214:              id: firstPage
215:              objectName: "Dashboard"
216:
217:              Dash{
218:                  id: dashObj
219:                  anchors.horizontalCenter: parent.horizontalCenter
220:                  anchors.verticalCenter: parent.verticalCenter
221:                  context: main
222:              }
223:            }
224:
225:            Item {
226:              id: secondPage
227:              objectName: "Diagnostics"
228:
229:              Diagnostics{
230:                  id: diagObj
231:                  // Items.Button{
232:                  //   text: "Settings"
233:                  //   style: main.config['style']['current']
234:                  //   implicitWidth: 100
235:                  //   onClicked: stack.push(view2)
236:                  //   anchors.bottom: parent.bottom
237:                  // }
238:                  anchors.horizontalCenter: parent.horizontalCenter
239:                  anchors.verticalCenter: parent.verticalCenter
240:                  context: main
241:              }
242:            }
243:
244:          }
245:
246:          PageIndicator {
247:            id: indicator
248:
249:            count: swipeView.count
250:            currentIndex: swipeView.currentIndex
251:
252:            anchors.bottom: swipeView.bottom
253:            anchors.horizontalCenter: parent.horizontalCenter
254:          }
255:
256:
257:        }
258:      }
259:
260:      Component.onCompleted: function(){
261:
262:          if(style == null)
263:            return;
264:
265:          sendInfoExtended("Connecting...", function(){
266:            // console.log("connection-established: " + main.diagnostics['connecti
```

```
on-established’])
  267:             return main.diagnostics[’connection-established’]
  268:           });
  269:
  270:           if(main.handler[’dev’]){
  271:             console.log("DEV MODE ENABLED!")
  272:             printTest()
  273:           }
  274:
  275:
  276:         stategroup.state = style;
  277:
  278:       }
  279:
  280:       StateGroup {
  281:         id: stategroup
  282:         states: [
  283:             State {
  284:               name: "dark"
  285:               PropertyChanges { target: root; color: "#000123"}
  286:             },
  287:             State {
  288:               name: "light"
  289:               PropertyChanges { target: root; color: "white"}
  290:             }
  291:         ]
  292:       }
  293:
  294:
  295:
  296:
  297:
  298:
  299:     }
  300:
  301: }
```