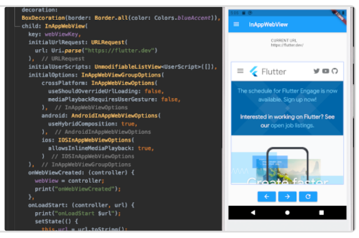
Blockly

- Stateful Widget to display the Blockly integration.
- Comprises of InAppWebViewer and Toolbox Buttons.

Packages used:

1. InAppWebView

Home
Adding the InAppWebView widget into your app is very simple. It's just a widget like any other Flutter widget:
https://inappwebview.dev/



2. flutter_svg

flutter_svg | Flutter Package
Draw SVG (and some Android VectorDrawable (XML)) files on a Flutter Widget. This is a Dart-native rendering
https://pub.dev/packages/flutter_svg

pub.dev

3. share

share | Flutter Package
This plugin has been replaced by the Flutter Community Plus Plugins version,share_plus . No further updates
https://pub.dev/packages/share

pub.dev

Modular Widgets used:

1. *ToolboxButtons - gblox_mobile\lib\components\Modular_Widgets\ToolboxButton*
2. *Button - gblox_mobile\lib\components\Modular_Widgets\Button*
3. *Cards - gblox_mobile\lib\components\Modular_Widgets\Cards*

Pages Used:

1. SaveProject - *gblox_mobile\lib\components\Blockly\save_project.dart*
2. OpenProject - *gblox_mobile\lib\components\Blockly\open_project.dart*

InAppWebView

- The major contributor of the blockly integration, this widget is used to load in an html page containing a simpler version of the gBlox interface. This is in the form of an html page, developed using react. The react project files are located in `blockly_react`, with majority of the code taken from the desktop application. The only things omitted are the toolbox, Pull out menu and headers.
- Steps for editing blockly in the application:
  1. Navigate into the `blockly_react` folder and run `npm run start`.
  2. Make edits as necessary using the knowledge from the desktop application.
  3. Any variables/functions needed to be used by flutter can be assigned to the window. e.g `window.currentWorkspace = currentWorkspace`
  4. Build the application by running `npm run build`.
  5. Copy the contents of `blockly_react/build` into `assets/react-blockly`.
  6. Restart flutter (not reload as it needs to rebuild the application with the new assets) and check the code section.
- Upon loading the blockly webpage, `react-blockly` `console.logs` necessary information required for a clean flutter integration. These console logs include:
  - toolbox data - a list of toolbox data required to generate flutter buttons.
  - xml data - blockly xml data for the current workspace used for saving `.gbx` files.
  - code data - code generated by the blocks used to upload to the robot via Bluetooth
  - variables created - all blockly variables generated also used for saving `.gbx` files
- `InAppWebView` also changes the device to the global device set by the user previously. It uses `document.getElementbyId("${ global.selectedDevice} ").click()` to simulate a click on the device from the hidden device manager component in `react-blockly`.
- A `webController` is stored as a global variable to be used by the `ToolBoxButtons`. Methods for the `webController` can be found on the package page on flutter, found at the beginning of this chapter.

Toolbox Generator

- From the time that `InAppWebView` receives the toolbox data, it is processed and stored in `List<List<String>> toolbox`. The function `updateToolbox()` is called and generates the widgets for the toolbox buttons. It sets the loaded value to true, which triggers a `ValueListenableBuilder` to build the new toolbox widget.

Saving and Loading Files

Saving Files

```
GBloxButtons(
                        buttonType: "menuButtons",
                        buttonName: "Save",
                        pressed: () async {
                          await controller!.evaluateJavascript(source: '''
                          console.log("xml: " + window.mainBlockly.Xml.domToText(window.mainBlockly.Xml.workspaceToDom(window.currentWorkspace)))
                          ''').then((value) async {
                            if (saveinfo.filename.isEmpty) {
                              saveinfo = global.SaveInformation(
                                  currentXml.value,
                                  global.selectedDevice,
                                  variables,
                                  "",
                                  "",
                                  false);
                              var data = await Navigator.push(
                                  context,
                                  MaterialPageRoute(
                                      builder: (context) => SaveProject(
                                          saveAs: true, saveData: saveinfo)));
                              setState(() {
                                saveinfo = data;
                              });
                            } else {
                              saveinfo.xml = currentXml.value;
                              saveinfo.variables = variables;
                              saveinfo.device = global.selectedDevice;
                              var data = await Navigator.push(
                                  context,
                                  MaterialPageRoute(
                                      builder: (context) => SaveProject(
                                          saveAs: false, saveData: saveinfo)));
                              setState(() {
                                saveinfo = data;
                              });
                            }
                          });
                        },
                      ),
```

- Saving of `.gbx` files require:
    - Xml data from blockly
    - Variable data from blockly
    - Device used to code
    - File Path if file was previously saved/loaded.
    - File Name if file was previously saved/loaded.
- The `webController` requests for the current workspace xml to be console logged, which will be set to `currentXml` variable. This is sent in with the data required to save the file and return the file path and name.
- This data is sent to `SaveProject`, with the presence of a saved/loaded file determining if the file is Overwritten or Saved as a new file.

Opening Files

```
GBloxButtons(
                        buttonType: "menuButtons",
                        buttonName: "Open",
                        pressed: () async {
                          var data = await Navigator.push(
                              context,
                              MaterialPageRoute(
                                  builder: (context) => OpenProject(
                                        fromHome: false,
                                      )));
                          try {
                            setState(() {
                              saveinfo = data;
                            });
                            global.selectedDevice = saveinfo.device;
                            variables = saveinfo.variables.toString();
                            currentXml.value = saveinfo.xml.toString();
                            controller!.evaluateJavascript(
                                source:
                                    '''loadBlocklyVariables("${variables}")''');
                            controller!.evaluateJavascript(source: '''
                                  var xml = window.mainBlockly.Xml.textToDom('${currentXml.value}');
                                  window.mainBlockly.Xml.clearWorkspaceAndLoadFromXml(xml,window.currentWorkspace)
                          ''').then((response) {
                              controller!.evaluateJavascript(source: '''
                                  var xml = window.mainBlockly.Xml.textToDom('${currentXml.value}');
                                  window.mainBlockly.Xml.clearWorkspaceAndLoadFromXml(xml,window.currentWorkspace)
                          ''');
                            });
                          } catch (e) {}
                        },
                      ),
```

- Opening `.gbx` files is done on a separate page, with the internal files displayed on the page. The user also has the option of choosing a file from the phone, one that may have been shared by another user. On choosing the file, the data is sent back and stored. Variables are loaded first with the workspace loaded right after. Due to possible lagging of the loading of variables, the blockly xml is loaded twice.