Robotics-GUI Documentation

Release 0.2.0

Angus Finch

Jan 28, 2020

Welcome

1 Contributing 3

2 Quick start 5

3 Introduction to Electron 7
- 3.1 What is Electron? 7
- 3.2 Why use Electron? 7
- 3.3 Examples of Electron 7

4 Getting the robot running without any fuss 9
- 4.1 Installation 9
- 4.2 Getting the code on the Pi 10
- 4.3 Starting up 10

5 Installing the project 11
- 5.1 Installing NodeJS 11
- 5.2 Electron guides 11
- 5.3 Installation of the GUI 11
- 5.4 Installation breakdown 12

6 Code structure 13

7 ./Main.js 15

8 ./Server/server.js & ./App/index.html 17

RoboHUD is a GUI created for the King's Legacy robotics team for use in the RoboCup international competition that takes place yearly

If you want to get straight to the code, it is available on GitHub

This documentation aims to cover the following:

- What RoboHUD is
- How to install all of the relevant tools in order to develop the GUI
- How the code works

If you have any questions at any point, feel free to contact Angus Finch via email, in the CCGS Robotics Discord server or in TG11 most days after school. You can also have a conversation with Mr. Nolan or one of the senior members for some help.

Next up: an introduction to Electron

Documentation tree:

CHAPTER 1

Contributing

If you want to open an issue, please do so here.

If you would like to contribute to the GUI, please talk to Angus Finch (Any help greatly appreciated)

CHAPTER 2

Quick start

If you have experience with NodeJS and NPM, and have all of the relevant binaries installed, get started quickly using:

```
git clone https://github.com/Finchiedev/robotics-gui.git
cd robotics-gui
npm install -y
npm start
```

CHAPTER 3

Introduction to Electron

3.1 What is Electron?

Electron is a fork of Google's Chromium project that allows for an open source tool to create native(ish) applications. It uses the same web technologies from the Chrome browser and combines it with the diverse range of platforms Chrome is available on to create a cross-platform, easy to use development experience. It also opens up unlimited possibilities by using Node.js to provide thousands of packages with endless uses.

3.2 Why use Electron?

As stated previously, Electron allows for cross-platform builds and a large collection of packages. We can use it to take advantage of:

- The ability to run builds on the student's Surface Books
- The built-in network tools to communicate easily, efficiently and quickly with the pi
- Web technologies that have been developed for online use
- NodeJS packages that allow for extended funtionality

3.3 Examples of Electron

Electron might be more common than you think. Don't believe me? Check out some popular examples:

- Discord
- GitHub's "Atom" text editor
- Skype
- GitHub's "Desktop" Git client

- Axosoft’s “GitKraken” Git client
- Hyper Terminal
- WordPress
- Visual Studio Code
- Slack

CHAPTER 4

Getting the robot running without any fuss

Do you want to just get the robot to work? This is the place! This page will show you how to get the GUI going in (hopefully) under 10 minutes. Please note - at the time of writing, the GUI code is not modular. This means it will currently not work for any robot you turn on without some tweaking. If you **do** need help, please come find Angus Finch, a senior member of the robotics club or Mr. Nolan/Felix.

4.1 Installation

First things first, you must be using an linux laptop.

Next, open Terminal by typing `terminal` into the search bar.

Next make sure you have installed NodeJS and NPM:

```
node -v && npm -v
```

Note if this command returns something like:

```
Command 'node' not found, but can be installed with:
```

```
sudo apt install nodejs
```

You may install it using:

```
sudo apt install nodejs
```

From there, we are able to install the RoboHUD project (make sure that you are in the directory you want to work in):

```
git clone https://github.com/CCGSRobotics/RoboHUD.git
```

CD into the directory so we are in the right place:

```
cd RoboHUD/
```

Install all of the required packages with the following (this will install in the top directory and install in the server directory):

```
npm install && cd Server/ && npm install && cd ..
```

4.2 Getting the code on the Pi

Now this is a tricky set of commands, but it is very useful to transfer files to the robot without any internet on the robot. Change to the robot's wireless network, it should look something like `KingsLegacy` Open a new terminal window and SSH into the Pi:

```
ssh pi@192.168.100.1
```

Check if there is an existing GUI folder in ~/:

```
ls | grep "GUI"
```

If there is no output from that command, run the following:

```
mkdir GUI
```

From there, you can CD into the GUI directory:

```
cd GUI
```

There is a **very** neat tool called scp, which uses the SSH technology to copy files across devices. Try using the following in the original terminal you opened:

```
scp -r Server/ pi@192.168.100.1:~/GUI/
```

4.3 Starting up

Go back to the SSH terminal and type in `ls` to make sure the files have been copied across. There should be a Server/ folder with some files in it.

Once again, use `cd Server/` to go into the project directory.

Use the command `node server.js` to start the compatability server

Return to the original terminal and run `npm start`. The GUI should start, with the only thing left do do being the addition of a controller!

CHAPTER 5

Installing the project

5.1 Installing NodeJS

It is a good idea to install NodeJS if you wish to follow along with this tutorial. All you have to do is go to the NodeJS downloads page and install the relevant files. Then follow the instructions to finish setup. Check you have the language installed by typing the following into a shell:

```
node -v
npm -v
```

If there are numbers appearing underneath the commands, NodeJS and NPM have been installed. Now to use them!

5.2 Electron guides

Electron has a quick start guide that demos how to get a simple Electron app running. To get started, simply type the following commands:

```
git clone https://github.com/electron/electron-quick-start.git
cd electron-quick-start
npm install
npm start
```

There is also a more extensive demo, which includes some things that Electron can do to imitate native apps. This can be found at Electron API demos, but this documentation will not cover it in detail.

5.3 Installation of the GUI

By this point you would have poked around the very primitive Electron demo. Or maybe not. Your choice. To install the robotics GUI, use the following commands:

```
git clone https://github.com/Finchiedev/robotics-gui.git
cd robotics-gui
npm install
```

Before starting the application, make sure that you have Python installed. Chances are, if you're reading this guide & following along, you have a Linux distro booted. If you have no idea what that is, please ask someone to help you with setting up a laptop. If you would rather use another OS, the installation of Python is relatively easy. **On Windows:** Simply go to the Python downloads page and find the relevant Windows installer. Run the .exe file, and make sure that the box labelled *Add Python to your PATH* IS CHECKED. This is incredibly important for later Done! Congratulations on installing Python on your machine!

On MacOS: Tricked you! Python3 should be installed by default. Simply go to spotlight, type in terminal and check that Python is installed by typing:

```
python3
```

If it works, there should be a sign that looks like **>>>**. Type quit() to exit.

5.4 Installation breakdown

Above you were shown the basic workflow for installing an Electron application. Let's break it down.

The first part of the code uses Git:

```
git clone <project url>.git
```

This gives your command line the following instructions:

1. Use package **Git** to complete this operation
2. The function for Git to perform is to clone (Download) a project
3. Git should download from the specified URL and make it into a folder called <project-name>

I highly recommend learning more about Git with the "Pro Git" book. Even if you have used before, it is worth reading, as you are bound to learn something! It also helps develop skills for a real-world professional programming environment

The next commands are relatively simple:

- cd <project-name> [Tells the computer to go to that folder]
- npm install [Uses NPM (*Node Package Manager*) to install the required files]
- npm start [Instructs NPM to run the start script]

CHAPTER 6

Code structure

The structural rules for the code are simple, and go as follows:

```
README.md is for users to get started
main.js is for the Electron renderer process
package.json & package-lock.json are for NPM use
.travis.yml & .circleci/ are for CI
App/*.html is for all HTML files
App/CSS/*.css is for all CSS files used by the HTML pages
App/JS/*.js is for all JavaScript files used by the HTML pages
App/JS/Resources/ is for NodeJS resources
```

CHAPTER 7

./Main.js

This is the first file that the Electron program runs when you execute **npm start**. Let’s go through it, piece by piece.

```
const {
app,
BrowserWindow
} = require('electron')
```

The purpose of the code above is to make sure that NodeJS knows what Electron is when it starts up. It calls app and BrowserWindow as “constructors”, used by NodeJS to integrate packages. This is the equivalent of the Python3 code:

```
import app, BrowserWindow from Electron
```

The rest of the code sets up a new BrowserWindow with the required specifications, adding listeners to figure out what to do on certain events.

CHAPTER 8

./Server/server.js & ./App/index.html

This is the main command layer that talks to the GUI and allows for a better user experience, allowing for automated, human-readable commands to be run. When you press the start button on the GUI, this is what follows:

- The client writes *VIDEO* to the server
- The server recognises this command and executes the required camera streaming code
- After 1 second to allow time to execute, the client writes *START* to the server
- The server recognises this command and executes the required server code
- After 3 seconds to allow time to execute, the client begins to write servo instructions