User Manual

By Scott Miller

Table of Contents

[Hardware Checklist 3](#_Toc140500018)

[Arduino Mega 2560 Rev3 3](#_Toc140500019)

[Custom PCB 3](#_Toc140500020)

[Med Associates Devices 3](#_Toc140500021)

[Software Checklist 3](#_Toc140500022)

[Client Application 3](#_Toc140500023)

[Angular Pre-Installations 3](#_Toc140500024)

[Creating a New Application 3](#_Toc140500025)

[Angular Package Installations 4](#_Toc140500026)

[Angular Material 4](#_Toc140500027)

[Bootstrap 5](#_Toc140500028)

[Additional Angular CLI Commands 6](#_Toc140500029)

[RESTful API 6](#_Toc140500030)

[Python Installation 6](#_Toc140500031)

[Python Modules Installation 6](#_Toc140500032)

[Virtual Environment 6](#_Toc140500033)

[Installing pip Modules 7](#_Toc140500034)

[Arduino IDE 7](#_Toc140500035)

[Angular GUI 7](#_Toc140500036)

[Operation Procedure 8](#_Toc140500037)

[Maintenance 8](#_Toc140500038)

# Hardware Checklist

[something summarizing this whole section, nothing long]

## Custom PCB

A custom printed circuit board (PCB) was created for this project, which is shown in Figure 1. Schematic and board files are included in the project design files, which were made in Autodesk EAGLE. Gerber files, which are sent to manufacturers to produce, in the form of a zip file is included in the project design files as well in case there is no access to the schematic and board file viewers. A complete connection overview is shown in Figure 2, where connections with the Arduino Mega 2560 Rev3, the 25-pin serial cable that conducts digital logic of the Med Associates Inc operant chamber products, and alligator clips that power the operant chamber are visible.

A green circuit board with many wires

Description automatically generated

Figure : An image of the created PCB with appropriate jumper wires (left and top) and serial cable (left). Connections not shown is a power cord that plugs into the black housing on the bottom of the board. The two squares of arrays are irrelevant to the performance of the proposed system and was included in the design for a different project.

A circuit board with wires attached to it

Description automatically generated

Figure : []

## Arduino Mega 2560 Rev3

An Arduino Mega 2560 Rev3 was used for this project, some information of which is included below. Extended information can be found at <https://store-usa.arduino.cc/products/arduino-mega-2560-rev3?selectedStore=us>, which is the same resource that the below diagrams originate from.

A circuit board with many components

Description automatically generated

Figure : Pinout of the Arduino Mega 2560 Rev3 board.

The Arduino Mega 2560 Rev3 pinout can be segmented into three groups. The first group is the input connections from the custom PCB, which occupy pins 3 to 10 on the microcontroller. Input 1 of the PCB relates to pin 3 on the Arduino, input 8 of the PCB relates to pin 10 on the Arduino, and the channels between increase sequentially on the Arduino pins. The second group is the output connections from the custom PCB, which occupy the even numbered pins between, and including, pin 22 and pin 52 for a total of 16 pins. Output 1 of the PCB relates to pin 52 on the Arduino, output 16 of the PCB relates to pin 22 of the Arduino, and the channels between decrease sequentially on the even pins of the Arduino. The last group is the ground pin of the Arduino that connects to the ground node of the custom PCB. The pinout with jumper cables is shown in Figure X.

A circuit board with wires

Description automatically generated

Figure : Image of the jumper cables (top and right) plugged into the Arduino Mega Rev3. Connections not shown is a USB connector that plugs into the silver housing to the top-left of the board.

To run the code given in the project design files, the Arduino Mega 2560 Rev3 should be flashed with the Arduino code called [software name]. If the appropriate connections are made, then the Arduino code should declare and work with the appropriate pins for the system to operate.

## Med Associates Devices

[Thingies!]

# Software Checklist

The following sections are included to summarize the installation and troubleshooting of softwares used within the entire proposed system, from the client application to the representative state transfer (REST or RESTful) application programming interface (API). [could say more]

## Client Application

The following subsections outline relevant packages and installation procedures for software used within the client application of this project that allow it to be built and served. Relevant external links, steps, and more are included to assist with the installation and establishment of the workspace required to run Angular applications, but if additional problems are encountered then alternative resources may need to be sought out.

### Angular Pre-Installations

Before you can run and create Angular applications, there are some prerequisites necessary for the code to build and serve. The latest version of Node.js can be downloaded from their official website (<https://nodejs.org/en/download/>), and the version used in this project can be found in Table Y2. The latest version of Git can be downloaded from their official website (<https://git-scm.com/downloads>), and the version used in this project can be found in Table Y2. Once those are installed, appropriate commands from the npm package manager can be executed within the command terminal to download the Angular CLI (<https://angular.io/guide/setup-local>) with the following command:

npm install -g @angular/cli

The version of the Angular CLI used in this project can be found in Table Y2. Default settings were accepted in each software’s installation, and administration permissions were not required from my limited experience. These softwares must be installed to build and serve Angular applications.

#### Creating a New Application

If there is interest in creating a new Angular application project, then the following steps can be followed to create an application. A default web application called example-app will be installed from these steps, but the code can be easily replaced and customized – even the HTML file acknowledges this in included comments. These steps were adapted from the official Angular website (<https://angular.io/guide/setup-local>).

1. Open a command terminal on your computer.
2. Navigate to a directory to download and configure a project folder within.
3. Type the following command into the terminal, and hit enter:

ng new example-app

1. If asked, type “y” or “N” to indicate whether you want to share pseudonymous data with the Angular Team at Google.
2. Type “y” or “N” to indicate whether you want to use Angular routing, more information can be found here: <https://angular.io/guide/routing-overview>.
3. Use arrow keys to change the selection of stylesheet format, and use enter to make the selection. CSS is commonly used.
4. Several files will be created in a new folder within the directory, and packages will begin installing. Once finished, each file might have the following warning indicator: “LF will be replaced by CRLF the next time Git touches it”, but it should be fine.

### Angular Package Installations

There are many packages that can be used to create an appealing and stunning UI, but in this project only two packages were used in their simplest form: Angular Material UI Components and Bootstrap. The version of both packages used in this project are indicated in Table Y2.

#### Angular Material

There are two ways that you can download and add Angular Material to a new project file. Both methods should allow for the implementation of these UI components to facilitate form fields and more. The first method creates a folder called “node\_modules” in whatever directory it was downloaded in from the command terminal, which acts as either a public or private npm registry for Angular applications to use (<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>). In later versions of the npm package manager, such as used in this project, the “-- save" is no longer necessary (<https://stackoverflow.com/questions/19578796/what-is-the-save-option-for-npm-install>), but I wanted to include the exact commands I used in my original downloads. The second method directly adds the Angular Material packages into the “node\_modules” folder that is present within the Angular project directory, which just saves an extra registry being present elsewhere on the local hard drive.

Method 1:

1. Open a command terminal on your computer.
2. Navigate to another directory to download this folder of packages, if desired.
3. Type the following command into the terminal, and hit enter:

npm install --save @angular/material

1. Packages should be added and audited, as indicated by text generated within the command terminal. The following is a sample of what was generated when doing this command:

added 63 packages, and audited 64 packages in 11s

2 packages are looking for funding

run `npm fund` for details

found 0 vulnerabilities

Method 2:

1. Open a command terminal on your computer.
2. Navigate to the Angular project directory.
3. Type the following command into the terminal, and hit enter:

ng add @angular/material

1. Packages will be installed into a directory within the Angular project.
2. Use arrow keys to change the selection of a theme, and use enter to make your selection.
3. Type “y” or “N” to indicate whether you want or don’t want global typography in your Angular project, respectively. Indicating “y” does not result in an error.
4. Use arrow keys to change the selection of whether to include browser animations, and use enter to make your selection. This application does not use them, so they were not included.
5. HTML, JSON, and CSS files will be updated with your selections and the newly installed packages from step 4.

#### Bootstrap

Bootstrap is an old and popular means of shaping UI layout, and while it is not used in this current project it can be easily added with the following steps. These steps outline the addition of the CSS style to the Angular application, but not the script aspect. Please reference <https://www.ngdevelop.tech/bootstrap-installation/> for additional assistance and adding bootstrap script to the Angular project. A useful remark is that if any JSON files are updated while the application is running, then the application will need to be stopped and rebuilt so the JSON files can be re-read.

1. Open a command terminal on your computer.
2. Navigate to the Angular project directory.
3. Type the following command into the terminal, and hit enter:

npm install --save bootstrap

1. Packages will be installed into the “node\_modules” directory within a folder with the same name.
2. Open a software capable of editing JSON files, such as Microsoft Visual Studio, and open the angular.json file within your Angular project directory.
3. Find the styles property within the JSON file (“angular.JSON”), and add the file pathway to the “bootstrap.min.css” file within array. A sample of this file pathway, which may not be yours, is as follows:

"node\_modules/bootstrap/dist/css/bootstrap.min.css"

#### Additional Angular CLI Commands

There are a lot of commands and additional flags (indicated by a “--") that can streamline the coding process. These commands can be executed in a Developer PowerShell, like the one present in Microsoft Visual Studio, or in the command terminal within the Angular application’s directory. A summary of the commands are found at <https://angular.io/cli>, which routes to specific documentation such as <https://angular.io/cli/generate> and <https://angular.io/cli/serve> to indicate additional parameters and uses.

## RESTful API

The following subsections outline relevant modules and installation procedures for software used within the RESTful API to allow it to compile and run. Relevant external links, steps, and more are included to assist with the installation and establishment of the program and a virtual environment, but if additional problems are encountered then alternative resources may need to be sought out.

### Python Installation

The RESTful API was built out of modules made within the Python programming language. The most recent version of Python can be downloaded from their official website (<https://www.python.org/downloads/>), and the version used in this project can be found in Table Y2. An important remark is that Python indicates on its download page that it will no longer support Python 3.8 starting October 2024, so the RESTful API might need version updates following this scheduled date.

### Python Modules Installation

All modules included in this project were installed through the preferred installed program (pip), which is included in recent versions of Python (<https://docs.python.org/3/installing/index.html>). This resource offers troubleshooting assistance for ensuring that pip was installed correctly and some explanation of commands to install the modules.

#### Virtual Environment

This project uses, and comes with an already established, virtual environment to isolate the Python packages and programs relevant to the RESTful API from the version requirements present elsewhere on the computer. This is an advantage to the use of virtual environments and the reason why the RESTful API was designed within one. To use the established virtual environment, skip the following steps until the instructions on activating the virtual environment. To create a new virtual environment, follow the steps that were adapted from (<https://docs.python.org/3/tutorial/venv.html>) and (<https://docs.python.org/3/library/venv.html>).

1. Open a command terminal on your computer.
2. Navigate to a directory where the virtual environment dependencies, files, and more are desired to be stored in.
3. Type the following command into the terminal, and hit enter:

python -m venv .venv

1. A directory will be created in the directory the command terminal was navigated to, and it will have the name “.venv”.

Once the virtual environment has been created it must be activated before modules can be installed within it or software within to be executed. Virtual environments are temporarily activated through the command terminal, navigated to the directory containing the “.venv” directory, with the following command:

.venv\Scripts\activate.bat

When activated, any Python programs inside can be executed, and any modules installed through pip will be installed within the folders of the virtual environment. One indicator that the virtual environment was opened in the command terminal is if the name of the virtual environment appears in parentheses before the file pathway (i.e. (.venv) C:\Users\...). A useful command to create a text file that displays all the installed modules is as follows:

python -m pip freeze > requirements.txt

When the activated virtual environment is finished in use or needs to be shut down, the following command is used through the command terminal to deactivate the virtual environment:

deactivate

#### Installing pip Modules

The virtual environment that the RESTful API was created in should be equipped with all necessary Python modules, but if problems arise with version mismatching or if a desire to install these modules elsewhere appear then use the following command. If there is interest in isolating these modules within a virtual environment, then make sure that the virtual environment is active before executing the commands.

pip install <module>

Each following Python module will replace the <module> placeholder with a relevant identifier for pip to install the correct module. When installing Flask (<https://pypi.org/project/Flask/>) then include Flask instead of the placeholder. When installing Flask CORS (<https://flask-cors.readthedocs.io/en/latest/>) then include flask-cors instead of the placeholder. When installing PySerial (<https://pyserial.readthedocs.io/en/latest/pyserial.html#installation>) include pyserial instead of the placeholder. If necessary, freezing the software versions of the modules within the virtual environment in a text file might be useful for documentation purposes. The versions used for this project are indicated in Table Y2.

## Arduino IDE

The Arduino IDE is necessary to not only modify or generate code for the Arduino microcontrollers, but it is also necessary to flash finished code into the on-board memory of the Arduino microcontroller. The IDE can be installed from the official website (<https://www.arduino.cc/en/software>), and the version used is indicated in Table Y2; although, this software’s version be fine being the latest version.

# Angular GUI

[something summarizing this whole section, nothing long]

# Operation Procedure

[something summarizing this whole section, nothing long]

# Maintenance

[something summarizing this whole section, nothing long]

* Potential software version issues.
* Make sure URLs line up.
* Review relevant sections of the installation section to see if there are notes.
* Virtual environment variables:
  + setx FLASK\_APP "application.py"
  + setx FLASK\_ENV "development"
  + https://phoenixnap.com/kb/windows-set-environment-variable

[Table Y2 – software versions]