# File: Main.py

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: main.py

Description: Implements a web page for the IoT Garage Door controller. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Include libraries and references

\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Library: flask

\* Author: Armin Ronacher

\* Source: http://flask.pocoo.org/

\* Version: 1.0.2

\* Description: Flask is a microframework for Python based on Werkzeug, Jinja 2

\* and good intentions. And before you ask: It's BSD licensed!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Library: Flask Socket IO extentions

\* Author: Miguel Grinberg

\* Source: https://flask-socketio.readthedocs.io/en/latest/

\* Version: n/a

\* Description: Flask-SocketIO gives Flask applications access to low latency

\* bi-directional communications between the clients and the

\* server. The client-side application can use any of the

\* SocketIO official clients libraries in Javascript, C++, Java

\* and Swift, or any compatible client to establish a permanent

\* connection to the server.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Library: Python JSON library

\* Author: Python Software Foundation

\* Source: https://docs.python.org/2/library/json.html

\* Version: 2.7

\* Description: Provides JSON encoding/decoding capabilities in Python

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Library: Python Socket Library

\* Author: Python Software Foundation

\* Source: https://docs.python.org/2/library/socket.html

\* Version: 2.7

\* Description: Provides networking sockets in Python.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Library: sqlite3 Database Library

\* Author: Python Software Foundation

\* Source: https://docs.python.org/2/library/sqlite3.html

\* Version: 2.7

\* Description: Provides SQL Lite database connectivity in python.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

**from** flask **import** Flask**,** render\_template

**from** flask\_socketio **import** SocketIO

**import** json

**import** socket

**import** sqlite3

app **=** Flask**(**\_\_name\_\_**)**

app**.**config**[**'SECRET\_KEY'**]** **=** 'vnkdjnfjknfl1232#'

socketio **=** SocketIO**(**app**)**

STATUS\_VALS **=** **[]**

SETTING\_VALS **=** **[]**

BUF\_STATUS\_VALS **=** **[{**'name'**:** 'lightStatus'**,** 'value'**:** **False},**

**{**'name'**:** 'doorStatus'**,** 'value'**:** **False},**

**{**'name'**:** 'alarmStatus'**,** 'value'**:** **False}]**

UDP\_IP **=** '192.168.1.155'

UDP\_PORT **=** 1234

UDP\_TIMEOUT **=** 1

@app.before\_first\_request

**def** init**():**

"""Initialization function to perform certain actions before a client can connect"""

loadLocalSettings**()**

@app.route**(**'/'**)**

**def** sessions**():**

"""Renders the root document (index.html)"""

**return** render\_template**(**'index.html'**)**

**def** messageReceived**(**methods**=[**'GET'**,** 'POST'**]):**

"""Callback method to indicate a successful message was received"""

**print(**'message was received!!!'**)**

@app.route**(**'/status'**)**

**def** status**():**

"""Redirects the /status URL to render status.html"""

**return** render\_template**(**'status.html'**)**

@app.route**(**'/settings'**)**

**def** settings**():**

"""Redirects the /settings URL to render settings.html"""

**return** render\_template**(**'settings.html'**)**

@socketio.on**(**'getStatus'**)**

**def** handle\_getStatus\_event**():**

"""Sends the current status values to the client"""

**global** STATUS\_VALS

socketio**.**emit**(**'updateStatus'**,** str**(**STATUS\_VALS**),** callback**=**messageReceived**)**

@socketio.on**(**'refreshStatus'**)**

**def** handle\_refreshStatus\_event**():**

"""Handles the refreshStatus even to fetch the status from the controller"""

receiveStatus**()**

**def** convert**(**input**):**

"""

from https://stackoverflow.com/questions/13101653/python-convert-complex-dictionary-of-strings-from-unicode-to-ascii

Converts unicode characters to ASCII to be compatible with the ESP32

:param input: object (dict, list) or unicode string to convert

:returns: converted ASCII string

"""

**if** isinstance**(**input**,** dict**):**

**return** dict**((**convert**(**key**),** convert**(**value**))** **for** key**,** value **in** input**.**iteritems**())**

**elif** isinstance**(**input**,** list**):**

**return** **[**convert**(**element**)** **for** element **in** input**]**

**elif** isinstance**(**input**,** unicode**):**

**return** input**.**encode**(**'utf-8'**)**

**else:**

**return** input

@socketio.on**(**'setStatus'**)**

**def** handle\_setStatus\_event**(**jStr**):**

"""

Handles the setStatus event to parse a JSON string and send it to the controller

:param jStr: JSON object to be parsed (format: list of dictionaries: {'name': <statusName>, 'value': <statusValue>})

"""

out **=** convert**(**jStr**)**

**for** setting **in** out**:**

name **=** setting**[**'name'**]**

val **=** setting**[**'value'**]**

setStatus**(**name**,** val**)**

sendStatus**()**

receiveStatus**()**

@socketio.on**(**'getSettings'**)**

**def** handle\_getSettings\_event**():**

"""Handles the getSettings event to send the current configured setting values to the client"""

**global** SETTING\_VALS

jStr **=** json**.**dumps**(**SETTING\_VALS**)**

socketio**.**emit**(**'updateSettings'**,** jStr**,** callback**=**messageReceived**)**

@socketio.on**(**'setSettings'**)**

**def** handle\_setSettings\_event**(**jStr**):**

"""

Handles the setSettings even to update the current configured settings and store them in the database

:param jStr: a JSON object (format: list of dictionaries: {'name': <statusName>, 'value': <statusValue>})

"""

**global** SETTING\_VALS

conn **=** sqlite3**.**connect**(**'data/settings.db'**)**

**print(**'Received settings request: ' **+** str**(**jStr**))**

c **=** conn**.**cursor**()**

**for** setting **in** jStr**:**

name **=** setting**[**'name'**]**

val **=** setting**[**'value'**]**

**print(**val **+** ' ' **+** name**)**

c**.**execute**(**'UPDATE settings SET value=? WHERE name=?'**,** **[**val**,** name**])**

conn**.**commit**()**

c**.**execute**(**'SELECT \* FROM settings'**)**

SETTING\_VALS **=** sqlToDictList**(**c**.**fetchall**())**

conn**.**close**()**

**def** sqlToDictList**(**results**):**

"""

Converts the output of an SQL query to a list of dictionaries

:param results: Output of a SQL query (i.e. from a 'fetchall' command)

:returns: A list of dictionaries containing the data from the query

"""

outList **=** **[]**

**for** r **in** results**:**

dictObj **=** **{}**

dictObj**[**'name'**]** **=** r**[**0**]**

dictObj**[**'value'**]** **=** r**[**1**]**

outList**.**append**(**dictObj**)**

**return** outList

**def** loadLocalSettings**():**

"""Loads local settings from the sqlite database"""

**global** SETTING\_VALS

conn **=** sqlite3**.**connect**(**'data/settings.db'**)**

c **=** conn**.**cursor**()**

c**.**execute**(**'SELECT \* FROM settings'**)**

valStr **=** c**.**fetchall**()**

SETTING\_VALS **=** sqlToDictList**(**valStr**)**

conn**.**close**()**

**def** updateLocalSettings**():**

"""Updates web server settings like the UDP IP address, port, and the timeout"""

**global** UDP\_IP

**global** UDP\_PORT

**global** UDP\_TIMEOUT

UDP\_IP **=** str**(**getSetting**(**'udpIP'**))**

UDP\_PORT **=** int**(**getSetting**(**'udpPort'**))**

UDP\_TIMEOUT **=** float**(**getSetting**(**'udpTimeout'**))**

**def** getSetting**(**name**):**

"""

Returns the value of a specified setting

:param name: string containing the name of the desired setting

:returns: value of the desired setting

"""

**global** SETTING\_VALS

**for** val **in** SETTING\_VALS**:**

**if** val**[**'name'**]** **==** name**:**

**return** val**[**'value'**]**

**return** **None**

**def** setStatus**(**name**,** value**):**

"""

Adds status tuple to local buffer status values

:param name: Name of status to add

:param value: Value of named status

"""

**global** BUF\_STATUS\_VALS

**for** val **in** BUF\_STATUS\_VALS**:**

**if** val**[**'name'**]** **==** name**:**

val**[**'value'**]** **=** value

**else:**

val**[**'value'**]** **=** **False**

**def** receiveStatus**():**

"""Receives the current status from the controller and stores it in memory"""

**global** STATUS\_VALS

**global** UDP\_IP

**global** UDP\_PORT

**global** UDP\_TIMEOUT

sock **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_DGRAM**)**

sock**.**settimeout**(**UDP\_TIMEOUT**)**

mesg **=** '{"cmd": "getStatus", "arg": ""}'

**print(**'Sending message to ' **+** str**(**UDP\_IP**)** **+** ':' **+** str**(**UDP\_PORT**))**

sock**.**sendto**(**mesg**.**encode**(**'utf-8'**),** **(**UDP\_IP**,** UDP\_PORT**))**

sock**.**close**()**

**try:**

sock2 **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_DGRAM**)**

sock2**.**settimeout**(**UDP\_TIMEOUT**)**

sock2**.**bind**((**''**,** UDP\_PORT**))**

data**,** server **=** sock2**.**recvfrom**(**1024**)**

**print(**"data: " **+** str**(**data**)** **+** "\nserver: " **+** str**(**server**))**

STATUS\_VALS **=** data

**except** socket**.**timeout**:**

**print(**'REQUEST TIMED OUT!'**)**

**def** sendStatus**():**

"""Sends current stored status to the controller"""

**global** BUF\_STATUS\_VALS

**global** UDP\_IP

**global** UDP\_PORT

jStr **=** json**.**dumps**(**BUF\_STATUS\_VALS**)**

**print(**"sendStatus: " **+** str**(**BUF\_STATUS\_VALS**))**

**print(**'Sending message: "' **+** jStr **+** '" to IP: ' **+**

str**(**UDP\_IP**)** **+** ':' **+** str**(**UDP\_PORT**))**

bitWiseValue **=** 0

# convert to a bitwise operator:

**for** var **in** BUF\_STATUS\_VALS**:**

**if** var**[**'name'**]** **==** "alarmStatus"**:**

bitWiseValue **=** bitWiseValue **|** **(**0x1 **&** var**[**'value'**])**

**if** var**[**'name'**]** **==** "doorStatus"**:**

bitWiseValue **=** bitWiseValue **|** **(**var**[**'value'**]** **<<** 1**)**

**if** var**[**'name'**]** **==** "lightStatus"**:**

bitWiseValue **=** bitWiseValue **|** **(**var**[**'value'**]** **<<** 2**)**

cmdStr **=** '{CMD:' **+** chr**(**bitWiseValue**)** **+** '}'

sock **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_DGRAM**)**

sock**.**sendto**(**cmdStr**.**encode**(**'utf-8'**),** **(**UDP\_IP**,** UDP\_PORT**))**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

socketio**.**run**(**app**,** debug**=True,** port**=**5555**,** host**=**'0.0.0.0'**)**

# File: sqliteReadAll.py

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: sqliteReadAll.py

Description: Fetches all data from the database. Used for testing purposes.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

**import** sqlite3

conn **=** sqlite3**.**connect**(**'data/settings2.db'**)**

**def** main**():**

c **=** conn**.**cursor**()**

c**.**execute**(**'SELECT \* FROM settings'**)**

vals **=** c**.**fetchall**()**

**for** val **in** vals**:**

**print(**val**)**

conn**.**close**()**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

main**()**

# File: sqliteWrite.py

'''\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: sqliteWrite.py

Description: Writes data back to the database. Used for testing purposes.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*'''

**import** sqlite3

conn **=** sqlite3**.**connect**(**'data/settings.db'**)**

**def** main**():**

inName **=** ''

**while** **True:**

inName **=** input**(**'enter a name: '**)**

**if** inName **==** '(exit)'**:**

**break**

nameStr **=** inName

inName **=** input**(**'enter a value: '**)**

**if** inName **==** '(exit)'**:**

**break**

valStr **=** inName

params **=** **(**nameStr**,** valStr**)**

c **=** conn**.**cursor**()**

c**.**execute**(**'INSERT INTO settings VALUES ' **+** str**(**params**))**

conn**.**commit**()**

**print(**'Name: "' **+** nameStr **+** '", Value: "' **+**

valStr **+** '" - successfully added'**)**

conn**.**close**()**

**if** \_\_name\_\_ **==** '\_\_main\_\_'**:**

main**()**

# File: main.js

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: main.js

Description: Implements the main page functions for GarageRTC. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create socket

var socket **=** io**.**connect**(**"http://" **+** document**.**domain **+** ":" **+** location**.**port**);**

// On socket event "connect", emit "my event" with JSON object

socket**.**on**(**"connect"**,** function **()** **{**

socket**.**emit**(**"my event"**,** **{**

data**:** "User Connected"

**});**

// on form submission, capture username and message and emit a "my event" with the data in JSON

var form **=** $**(**"form"**).**on**(**"submit"**,** function **(**e**)** **{**

e**.**preventDefault**();**

let user\_name **=** $**(**"input.username"**).**val**();**

let user\_input **=** $**(**"input.message"**).**val**();**

socket**.**emit**(**"my event"**,** **{**

user\_name**:** user\_name**,**

message**:** user\_input

**});**

// Set focus on input.message

$**(**"input.message"**)**

**.**val**(**""**)**

**.**focus**();**

**});**

**});**

// On socket event "my response" (from python), remove "No message" and append the chat message

socket**.**on**(**"my response"**,** function **(**msg**)** **{**

console**.**log**(**msg**);**

**if** **(**typeof msg**.**user\_name **!==** "undefined"**)** **{**

$**(**"h3"**).**remove**();**

$**(**"div.message\_holder"**).**append**(**

'<div><b style="color: #000">' **+**

msg**.**user\_name **+**

"</b> " **+**

msg**.**message **+**

"</div>"

**);**

**}**

**});**

# File: settings.css

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: settings.css

Description: Style Sheets for the settings page functions for GarageRTC. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**.**validText **{**

**background-color: whitesmoke;**

**}**

**.**invalidText **{**

**background-color: palevioletred;**

**}**

**.**octicon **{**

**color: white;**

**}**

svg **{**

fill**: currentColor;**

**}**

# File: settings.html

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: settings.html

Description: Settings Page, used to allow users to tweak certain settings. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<!DOCTYPE html>

<html lang=**"en"**>

<head>

<title>**GarageRTC Settings**</title>

<meta charset=**"utf-8"**>

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**>

<!-- SocketIO (used for asynchronous callbacks) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.min.js"**></script>

<!-- jQuery (used to easily access elements and perform operations on them) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"**></script>

<!-- Bootstrap (used for making the UI look nice) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"**></script>

<link rel=**"stylesheet"** href=**"https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css"**>

<!-- Custom stylesheet for settings page -->

<link rel=**"stylesheet"** href=**"../static/settings.css"**>

</head>

<body>

<nav class=**"navbar navbar-inverse navbar-static-top"** role=**"navigation"**>

<div class=**"container"**>

<div class=**"navbar-header"**>

<button type=**"button"** class=**"navbar-toggle collapsed"** data-toggle=**"collapse"**

data-target=**"#bs-example-navbar-collapse-1"**>

<span class=**"sr-only"**>**Toggle navigation**</span>

<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

</button>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>

<ul class=**"nav navbar-nav"**>

<li><a href=**"{{ url\_for('sessions') }}"**>**Home**</a></li>

<li><a href=**"{{ url\_for('status') }}"**>**Status**</a></li>

<li><a href=**"#"**>**Settings**</a></li>

</ul>

</div>

</div>

</nav>

<div id=**"settingsContainer"** class=**"container-fluid"**>

<p id=**"errorMsg"**></p>

<form id=**"settingsForm"** action=**"javascript: submitForm()"**>

**Controller IP Address:**<br>

<input type=**"text"** name=**"udpIP"** id=**"udpIP"** class=**"validText"** placeholder=**"IP Address (e.g. 192.168.0.2)"**>

<input type=**"text"** name=**"udpPort"** id=**"udpPort"** class=**"validText"** style=**"width:80px;"** placeholder=**"Port"**><br>

**Screen Cycle Time:**<br>

<input type=**"text"** name=**"screenCycle"** id=**"screenCycle"** class=**"validText"**><br>

<br>

<input type=**"submit"** value=**"Save"**>

</form>

</div>

<script src=**"../static/settings.js"**></script>

</body>

</html>

# File: settings.js

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: settings.js

Description: Implements the settings page functions for GarageRTC. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create socket

var socket **=** io**.**connect**(**"http://" **+** document**.**domain **+** ":" **+** location**.**port**);**

var time **=** 0**;**

$**(**document**).**ready**(**function **()** **{**

socket**.**emit**(**"getSettings"**);**

**});**

// function to submit the settings form and send the data back to the server

function submitForm**()** **{**

var ipForm **=** document**.**getElementById**(**"udpIP"**);**

var ip **=** ipForm**.**value**;**

var sIP **=** ip**.**split**(**"."**);**

var isValid **=** sIP**.**length **==** 4**;**

**if** **(**isValid**)** **{**

**for** **(**var i **=** 0**;** i **<** 4**;** **++**i**)** **{**

var nIP **=** parseInt**(**sIP**[**i**]);**

**if** **(**nIP **>** 255 **||** nIP **<** 0**)** **{**

isValid **=** **false;**

**break;**

**}**

**}**

**}**

**if** **(**isValid**)** **{**

socket**.**emit**(**"setSettings"**,** **[{**"name" **:** "udpIP"**,** "value"**:** ip**}]);**

$**(**cycleTimeForm**).**addClass**(**"validText"**).**removeClass**(**"invalidText"**);**

document**.**getElementById**(**"errorMsg"**).**innerHTML **=** ""**;**

**}**

**else** **{**

$**(**ipForm**).**addClass**(**"invalidText"**).**removeClass**(**"validText"**);**

document**.**getElementById**(**"errorMsg"**).**innerHTML **=** "Please enter a valid IPv4 address!"**;**

**}**

var portForm **=** document**.**getElementById**(**"udpPort"**);**

var port **=** parseInt**(**portForm**.**value**);**

console**.**log**(**port**);**

**if** **(**port **>** 0 **&&** port **<** 65536**)** **{**

socket**.**emit**(**"setSettings"**,** **[{**"name" **:** "udpPort"**,** "value"**:** portForm**.**value**}]);**

$**(**cycleTimeForm**).**addClass**(**"validText"**).**removeClass**(**"invalidText"**);**

document**.**getElementById**(**"errorMsg"**).**innerHTML **=** ""**;**

**}**

**else** **{**

$**(**ipForm**).**addClass**(**"invalidText"**).**removeClass**(**"validText"**);**

document**.**getElementById**(**"errorMsg"**).**innerHTML **=** "Please enter a valid port!"**;**

**}**

var cycleTimeForm **=** document**.**getElementById**(**"screenCycle"**);**

var cycleTime **=** cycleTimeForm**.**value**;**

**if** **(!**isNaN**(**cycleTime**)** **&&** cycleTime**.**indexOf**(**'.'**)** **==** **-**1**)** **{**

socket**.**emit**(**"setSettings"**,** **[{**"name" **:** "screenCycle"**,** "value"**:** cycleTime**}]);**

$**(**cycleTimeForm**).**addClass**(**"validText"**).**removeClass**(**"invalidText"**);**

document**.**getElementById**(**"errorMsg"**).**innerHTML **=** ""**;**

**}**

**else** **{**

$**(**cycleTimeForm**).**addClass**(**"invalidText"**).**removeClass**(**"validText"**);**

document**.**getElementById**(**"errorMsg"**).**innerHTML **=** "Cycle time must be an integer!"**;**

**}**

**}**

// Handles the updateSettings event to change the values shown on screen to reflect the new settings

socket**.**on**(**"updateSettings"**,** function **(**msg**)** **{**

objs **=** JSON**.**parse**(**msg**);**

**for** **(**var i **=** 0**;** i **<** objs**.**length**;** **++**i**)** **{**

var itm **=** document**.**getElementById**(**objs**[**i**].**name**);**

**if** **(**itm **==** **null)** **{**

**continue;**

**}**

**switch** **(**itm**.**type**)** **{**

**case** "text"**:**

itm**.**value **=** objs**[**i**].**value**;**

**break;**

**case** "checkbox"**:**

itm**.**checked **=** objs**[**i**].**value**;**

**break;**

**case** "radio"**:**

var radios **=** document**.**getElementsByName**(**objs**[**i**].**name**);**

var ind **=** parseInt**(**objs**[**i**].**value**);**

radios**[**ind**].**checked **=** **true;**

**break;**

**}**

**}**

**});**

# File: Status.css

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: status.css

Description: Style Sheets for the status page functions for GarageRTC. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**.**toggleBtn **{**

**width: 80px;**

**}**

**.**toggleBtn\_ON **{**

**background-color: lime;**

**color: black;**

**}**

**.**toggleBtn\_OFF **{**

**background-color: red;**

**color: black;**

**}**

# File: status.html

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: status.html

Description: Status Page, used to allow users to view the current status and send

messages to the controller. See https://github.com/jharmer95/Garage-RTC/

for details on the Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<!DOCTYPE html>

<html lang=**"en"**>

<head>

<title>**GarageRTC Status**</title>

<meta charset=**"utf-8"**>

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**>

<!-- SocketIO (used for asynchronous callbacks) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.min.js"**></script>

<!-- jQuery (used to easily access elements and perform operations on them) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"**></script>

<!-- Bootstrap (used for making the UI look nice) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"**></script>

<link rel=**"stylesheet"** href=**"https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css"**>

<!-- Custom stylesheet for status page -->

<link href=**"../static/status.css"** rel=**"stylesheet"** type=**"text/css"**>

</head>

<body>

<nav class=**"navbar navbar-inverse navbar-static-top"** role=**"navigation"**>

<div class=**"container"**>

<div class=**"navbar-header"**>

<button type=**"button"** class=**"navbar-toggle collapsed"** data-toggle=**"collapse"**

data-target=**"#bs-example-navbar-collapse-1"**>

<span class=**"sr-only"**>**Toggle navigation**</span>

<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

</button>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>

<ul class=**"nav navbar-nav"**>

<li><a href=**"{{ url\_for('sessions') }}"**>**Home**</a></li>

<li><a href=**"#"**>**Status**</a></li>

<li><a href=**"{{ url\_for('settings') }}"**>**Settings**</a></li>

</ul>

</div>

</div>

</nav>

<div id=**"statusContainer"** class=**"container-fluid"**>

<div class=**"statusDiv"**>

<h3>**System Status**</h3>

<input type=**"button"** id=**"alarmStatus"** class=**"toggleBtn t\_alarm"** value=**"???"**>

</div>

<div class=**"statusDiv"**>

<h3>**Door Status**</h3>

<input type=**"button"** id=**"doorStatus"** class=**"toggleBtn t\_open\_closed"** value=**"???"**>

</div>

<div class=**"statusDiv"**>

<h3>**Light Status**</h3>

<input type=**"button"** id=**"lightStatus"** class=**"toggleBtn t\_on\_off"** value=**"???"**>

</div>

<div class=**"statusDiv"**>

<h3 id=**"tempStatus"**>**Temperature:**</h3>

<h4>**???**</h4>

</div>

<div class=**"statusDiv"**>

<h3 id=**"coStatus"**>**CO Level:**</h3>

<h4>**???**</h4>

</div>

</div>

<script src=**"../static/status.js"**></script>

</body>

</html>

# File: Status.js

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: status.js

Description: Implements the status page functions for GarageRTC. See

https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Create socket

var socket **=** io**.**connect**(**"http://" **+** document**.**domain **+** ":" **+** location**.**port**);**

var time **=** 0**;**

const getRate **=** 250**;**

const refreshRate **=** 5000**;**

// Triggers the getStatus event every x milliseconds

window**.**setInterval**(**function **()** **{**

socket**.**emit**(**"getStatus"**);**

**},** getRate**);**

// Triggers the refreshStatus event every x milliseconds

window**.**setInterval**(**function **()** **{**

socket**.**emit**(**"refreshStatus"**);**

**},** refreshRate**);**

// Adds the setStatus event to the click function of elements with the toggleBtn class

$**(**document**).**ready**(**function **()** **{**

$**(**".toggleBtn"**).**click**(**function **()** **{**

var tID **=** **this.**id**;**

socket**.**emit**(**"setStatus"**,** **[{** name**:** tID**,** value**:** **true** **}]);**

**});**

**});**

// Handles the updateStatus event and changes the elements based on the new

// status values received from the server

socket**.**on**(**"updateStatus"**,** function **(**msg**)** **{**

var objs**;**

**if** **(**typeof msg **!==** "string"**)** **{**

objs **=** JSON**.**parse**(**JSON**.**stringify**(**msg**));**

**}**

**else** **{**

objs **=** JSON**.**parse**(**msg**);**

**}**

**for** **(**var i **=** 0**;** i **<** objs**.**length**;** **++**i**)** **{**

var itm **=** document**.**getElementById**(**objs**[**i**].**name**);**

**switch** **(**itm**.**tagName**)** **{**

**case** "H3"**:**

var t **=** objs**[**i**].**value**.**toString**();**

itm**.**nextElementSibling**.**innerHTML **=** t**;**

**break;**

**case** "INPUT"**:**

console**.**log**(**"type: " **+** itm**.**type**);**

**if** **(**itm**.**type **===** "button"**)** **{**

**if** **(**$**(**itm**).**hasClass**(**"t\_alarm"**))** **{**

**if** **(**objs**[**i**].**value **==** "True"**)** **{**

$**(**itm**).**val**(**"ALARM"**);**

$**(**itm**).**addClass**(**"toggleBtn\_OFF"**).**removeClass**(**"toggleBtn\_ON"**);**

**}**

**else** **{**

$**(**itm**).**val**(**"OK"**);**

$**(**itm**).**addClass**(**"toggleBtn\_ON"**).**removeClass**(**"toggleBtn\_OFF"**);**

**}**

**}**

**else** **if** **(**$**(**itm**).**hasClass**(**"t\_on\_off"**))** **{**

**if** **(**objs**[**i**].**value **==** "ON"**)** **{**

$**(**itm**).**val**(**"ON"**);**

$**(**itm**).**addClass**(**"toggleBtn\_ON"**).**removeClass**(**"toggleBtn\_OFF"**);**

**}**

**else** **{**

$**(**itm**).**val**(**"OFF"**);**

$**(**itm**).**addClass**(**"toggleBtn\_OFF"**).**removeClass**(**"toggleBtn\_ON"**);**

**}**

**}**

**else** **if** **(**$**(**itm**).**hasClass**(**"t\_open\_closed"**))** **{**

**if** **(**objs**[**i**].**value **==** 0**)** **{**

$**(**itm**).**val**(**"OPEN"**);**

$**(**itm**).**addClass**(**"toggleBtn\_OFF"**).**removeClass**(**"toggleBtn\_ON"**);**

**}**

**else** **if** **(**objs**[**i**].**value **==** 2**)** **{**

$**(**itm**).**val**(**"STOP"**);**

$**(**itm**).**addClass**(**"toggleBtn\_OFF"**).**removeClass**(**"toggleBtn\_ON"**);**

**}**

**else** **if** **(**objs**[**i**].**value **==** 3**)** **{**

$**(**itm**).**val**(**"MOVE"**);**

$**(**itm**).**addClass**(**"toggleBtn\_OFF"**).**removeClass**(**"toggleBtn\_ON"**);**

**}**

**else** **{**

$**(**itm**).**val**(**"CLOSED"**);**

$**(**itm**).**addClass**(**"toggleBtn\_ON"**).**removeClass**(**"toggleBtn\_OFF"**);**

**}**

**}**

**}**

**break;**

**}**

**}**

**});**

# File: Index.html

<!--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

File: index.html

Description: Main Page, used as a start for users to click on their desired page.

See https://github.com/jharmer95/Garage-RTC/ for details on the

Open GarageRTC project.

Authors: Daniel Zajac, danzajac@umich.edu

Jackson Harmer, jharmer@umich.edu

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*-->

<!DOCTYPE html>

<html lang=**"en"**>

<head>

<title>**GarageRTC Home**</title>

<meta charset=**"utf-8"**>

<meta name=**"viewport"** content=**"width=device-width, initial-scale=1"**>

<!-- SocketIO (used for asynchronous callbacks) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/socket.io/1.7.3/socket.io.min.js"**></script>

<!-- jQuery (used to easily access elements and perform operations on them) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/jquery/2.1.3/jquery.min.js"**></script>

<!-- Bootstrap (used for making the UI look nice) -->

<script src=**"https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/js/bootstrap.min.js"**></script>

<link rel=**"stylesheet"** href=**"https://cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.3.7/css/bootstrap.min.css"**>

</head>

<body>

<nav class=**"navbar navbar-inverse navbar-static-top"** role=**"navigation"**>

<div class=**"container"**>

<div class=**"navbar-header"**>

<button type=**"button"** class=**"navbar-toggle collapsed"** data-toggle=**"collapse"**

data-target=**"#bs-example-navbar-collapse-1"**>

<span class=**"sr-only"**>**Toggle navigation**</span>

<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

<span class=**"icon-bar"**></span>

</button>

</div>

<!-- Collect the nav links, forms, and other content for toggling -->

<div class=**"collapse navbar-collapse"** id=**"bs-example-navbar-collapse-1"**>

<ul class=**"nav navbar-nav"**>

<li><a href=**"#"**>**Home**</a></li>

<li><a href=**"{{ url\_for('status') }}"**>**Status**</a></li>

<li><a href=**"{{ url\_for('settings') }}"**>**Settings**</a></li>

</ul>

</div>

</div>

</nav>

<h2>**Welcome to GarageRTC!**</h2>

<p>**Please click on a link in the navbar above.**</p>

</body>

</html>